DIPLOMAMUNKA

Perge Frigyes

Debrecen

2010

Debreceni Egyetem

Informatika Kar

A szemantikus web és kapcsolata XML alapú rendszerekkel

Témavezető:

Dr. Adamkó Attila

Egyetemi adjunktus

Készítette:

Perge Frigyes

Programtervező matematikus

Debrecen

2010

Tartalom

1 Bevezetés

1.1 A világháló napjainkban

Jelenleg a Weben található információk természetes nyelveken (magyar, angol, német, stb.), képek, video- és hanganyagok formájában állnak rendelkezésre. Ezek elsősorban emberek számára készültek, tehát az embereknek általában nem okoz gondot azok feldolgozása. Természetes módon tudnak a különböző forrásokból származó részinformációkból bonyolultabb összefüggéseket alkotni, tényekre következtetni.

A gépek azonban nem ilyenek, hanem alapvetően buták. A részinformációk haszontalanok a számukra, nehézkes az adatok kombinálása, analógiák megtalálása, és nem képesek a multimédiás anyagokon, például képeken található információk értelmezésére sem. Ezek a hiányosságok képezik a mai keresők problémáit is. A jelenlegi keresőóriások nagy erőfeszítéseket tettek a minél jobb keresési találatok érdekében, ám ezek a keresők statisztikai algoritmusokat használnak, emiatt nem képesek sem az információk megértésére, sem az azokon való következtetésekre, túl sok hamis találatot eredményeznek. Az információk jelentős része elérhetetlen a mai szöveg alapú keresők számára, ezeket nevezzük a *mély web*nek (deep web). A mély web kifejezés két dolgot takar. Egyrészt a Weben keresztül csak lekérdezéssel elérhető adatbázisok, valamint a nem szöveges formában adott dokumentumok.

Ezekre a problémákra megoldást jelenthetne, ha az adatforrásokhoz valamilyen módon további leírást lehetne kapcsolni. Ez a szemantikus web alapgondolata.

Diplomamunkám célja hogy bemutassam a „szemantikus web elképzelést”, a hozzá kapcsolódó technológiákat, valamint felhasználási lehetőségeit.

1.2 A metaadatokról

Metaadatnak vagy metainformációnak nevezzük az olyan információt, mely más információs forrásról nyújt leírást. A metaadatok használata univerzális eszköz arra, hogy jelentést társítsunk webes tartalmakhoz. Metaadat lehet például egy zenei szám szerzője, a hossza, megjelenésének dátuma vagy, hogy melyik albumon található. Ez nem egy új dolog, eddig is léteztek különböző metaadatok, gondoljunk például a HTML META nevű tegjére, vagy egy word dokumentumban is megtalálható szerző, tárgy, cím, állapot tulajdonságokra, melyek

mind-mind metainformációk. A szemantikus web is ezek használatán alapszik, másrészt ezen metaadatok segítségével következtetni is kell tudni az adott dokumentum tartalmára.

Ahhoz hogy a gép következtetni tudjon, a metaadatokat a gép által értelmezhető formátumúvá kell tenni. Mire van ehhez szükség?

- Az erőforrások egyértelmű elvezésére.
- Az adatok összekapcsolására, leírására szolgáló általános modellre.
- Módszer az adatok modell alapján történő elérésére.
- Közös szókészlet definícióra.
- Következtetési módszerekre.

Leegyszerűsített formában ezek az összetevők adják a szemantikus webet.

1.3 A szemantikus web

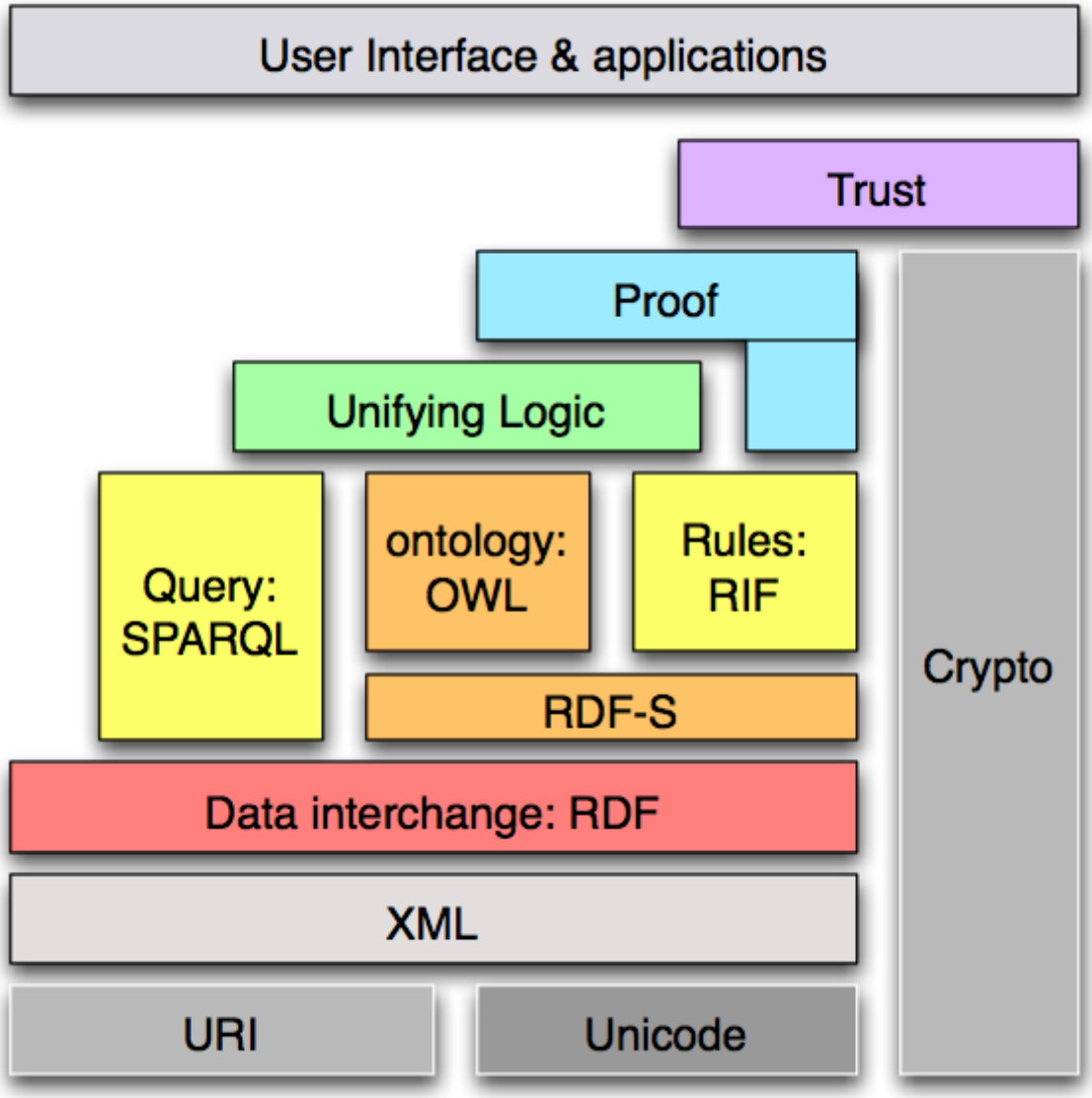
Most hogy megismerhettük a szemantikus web alapgondolatát és néhány alapelemét, szeretném pontosabban megfogalmazni mi is az a szemantikus web, mi a célja, és milyen elemekből épül fel.

A szemantikus web célja egy olyan infrastruktúra létrehozása, amely lehetővé teszi, hogy a gépek automatikusan tudják végezni, a Weben lévő adatok integrálását, a közöttük levő kapcsolatok definiálását és jellemzését, illetve az adatok értelmezését. A szemantikus web elgondolás Tim Berners-Lee-től származik, akit az egész World Wide Web és a hozzá kapcsolódó technológiák „atyjaként” is emlegetnek. Az ötletét 2000-ben tárta a nyilvánosság elé. Azóta a W3C szabványai közé tartozik, a jelenlegi világháló kiterjesztésének tekinthető.

A szemantikus web egymásra épülő rétegekből épül fel, mely fejlesztési folyamatnak is tekinthető. Ezt a réteges architektúrát szemlélteti az 1. ábra.

A szemantikus web rétegei

Unicode: A Unicode a különböző írásrendszerek egységes kódolását és használatát leíró ISO nemzetközi szabvány. A Unicode igyekszik magába foglalni a világ nyelveiben található összes előforduló karaktert, lehetővé téve a dokumentumok egész világon történő terjesztését.



1. Ábra: A szemantikus web rétegei (The Semantic Web Stack).

(forrás: http://www.w3.org/DesignIssues/diagrams/sweb-stack/2006a.png)

URI: Az URI (**U**niversal **R**esource **I**dentifier) az egységes erőforrás azonosítót takarja. A szókészlet alapja. Az erőforrások egyértelmű azonosítására szolgál.

XML: Az XML (E**X**tensible **M**arkup **L**anguage) elsődleges célja strukturált szöveg és információ megosztása az Interneten keresztül, számítógép számára is feldolgozható formában.

RDF: Az RDF (**R**esource **D**escription **F**ramework) az XML szintaxisára épülő keretrendszer a Weben történő információábrázolás céljára.

RDFS: Az RDF Séma (RDF Schema) egy szókészlet leíró nyelv, a tulajdonságok és osztályok leírására RDF–alapú forrásokhoz.

SPARQL: Egy modell alapú lekérdezőnyelv az RDF alapú adatok eléréséhez.

OWL: A szemantikus web számára az *ontológia* egy dokumentumot vagy egy fájlt jelent, amelyben formálisan definiálva vannak egy szakterület fogalmai és kapcsolatai. Az ontológiák leírására ontológianyelveket használhatunk. Az ontológianyelvek az RDF sémából származnak, annak kibővítéseként alkották meg őket. A jelenlegi hivatalos W3C ontológianyelv az OWL (**W**eb **O**ntology **L**anguage).

RIF: A RIF (**R**ule **I**nterchange **F**ormat) a szemantikus web „szabály rétege”.

Unifying Logic: A *leíró logikák* segítségével leírható valamely szakterület, vagy közismereti terület fogalmainak rendszere. A leíró logikák az elsőrendű logika résznyelvei, és ezek képezik az OWL hátterét. Amikor az ontológiákat ténylegesen működésbe akarjuk hozni, akkor az ontológiában lévő állításokat át kell alakítani leíró logikai állításokká.

Proof: Az OWL-ben, leíró logikai nyelven megadott állítások helyességének bizonyítására is szükségünk van. Ennek eszköze lehet a PML (**P**roof **M**arkup **L**anguage), mely az RDF alapjain nyugszik, és egy bizonyítás leírására szolgál. Emellett arra is használható, hogy segítse a különböző következtetőrendszerek közötti automatikus információcserét. Ezzel lehetővé válik, hogy megjelöljék azokat a leírásokat, amelyek ugyanazt a fogalomrendszert írják le.

Trust: Szükség van a bizalomra. Ha a felhasználó nem bízik meg a szemantikus weben szerzett információban, az eddig felsorolt technológiák nem sokat érnek. Biztosítani kell, hogy bárki, aki használni akarja a szemantikus webet, biztos lehessen abban, hogy az információ, amit kap, bizonyítottan igaz. Ehhez úgynevezett digitális aláírásokkal kell ellátni a dokumentumokat, amelyek olyan titkosított adatok, amelyek egyértelműen azonosítják az információk küldőjét, és biztosítják a címzettek számára az adatok sértetlenségét.

2 XML

Az RDF szabvány XML-szintaxisának ismertetéséhez szükséges, hogy ismerjük az XML nyelvet. A következőkben ezt próbálom meg bemutatni, a számunkra fontos részeire koncentrálva.

2.1 Mi az XML?

Az XML (E**X**tensible **M**arkup **L**anguage, Kiterjeszthető Leíró Nyelv) a W3C ajánlása. Általános célú leíró nyelv, speciális célú leíró nyelvek létrehozására. Az elsődleges célja strukturált szöveg és információ megosztása az Interneten keresztül, számítógép számára is feldolgozható formában. Az XML-en alapuló nyelvek, mint például az RDF/XML, formális módon vannak leírva, így lehetővé téve a programok számára a dokumentumok módosítását és validálását, a formátum előzetes ismerete nélkül.

Az XML az SGML egyszerűsített részhalmaza, hasonlóan a HTML nyelvhez. Formai szempontból sok hasonlóság és különbség található az XML- és HTML-dokumentumok között, ám más-más célt szolgálnak. A HTML-t az adatokat megjelenítésére, az XML-t az adatok strukturálására, tárolására és továbbítására tervezték.

2.2 Az XML használatának előnyei

Az XML alkalmas adattovábbításra a következő tulajdonságai miatt. Egyszerű szöveg, ezért mind ember, mind gép számára olvasható formátum. Támogatja az Unicode-ot, ami lehetővé teszi bármely információ bármely emberi nyelven történő közlését. Képes a legtöbb általános számítástudományi adatstruktúra ábrázolására, például rekord, lista, fa. Öndokumentáló, struktúra- és mezőneveket ír le speciális értékekkel együtt. Szigorú szintaktikus és elemzési követelményeket támaszt, ami biztosítja, hogy a szükséges elemzési algoritmus egyszerű, hatékony és ellentmondásmentes maradjon.

Az XML-t gyakran használják dokumentumtárolási és feldolgozási formátumként, mind online mind offline módon, és több előnnyel is jár. Ilyenek például, hogy internetes szabványokon alapul, erőteljes, logikailag ellenőrizhető formátum, a hierarchikus struktúrája megfelel a legtöbb dokumentum típusnak, licencektől és korlátozásoktól mentes, platform-független.

2.3 Alapvető szintaxis

Egy XML-dokumentum szöveges állomány, információt tárol, de ezen kívül „nem csinál semmit”. A feldolgozó alkalmazásnak kell eldöntenie, mihez kezd a dokumentumban tárolt adatokkal. Egy XML-dokumentum legfontosabb részei az elemek és az attribútumok, ezek névütközéseinek elkerülésére szolgálnak a névterek.

2.3.1 XML-elemek

Egy XML-dokumentum az adatokat hierarchikus formában tárolja, azaz a dokumentum egy fát határoz meg. Egy XML-elem három részből épül fel: egy nyitó címkéből (teg), az adatból, és egy záró címkéből, például `<nev>adat</nev>`. Tartalmuk alapján létezik egyszerű, összetett, vegyes és üres elem. Az összetett elemnek vannak gyermek elemei, az egyszerű tartalma egy literál, míg a vegyes elem tartalmaz literált és más elemeket is. Az üres elemek csak nyitó és záró címkékből állnak, megjelölhetők üres elem (önlezáró) teggel, mint például az `<ÜresVagyok/>`. Ez megegyezik az `<ÜresVagyok></ÜresVagyok>` párossal. Az XML-ben nincsenek foglalt szavak, így egy elem neve tetszőleges lehet.

A fentiek szemléltetésére nézzük meg az alábbi példát:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<gyökérelem>
        <egyszerű>tartalom</egyszerű>
        <összetett>
                <üreselem/>
                <egyszerű2>tartalom</egyszerű2>
        </összetett>
        <vegyes>
                <nagy>T</nagy>artalom
        </vegyes>
</gyökérelem>
```

1. példa: XML elemek

A dokumentumnak pontosan egy gyökérelemet kell tartalmaznia, esetünkben ez a `<gyökérelem>`. A gyökérelem gyermekei, illetve egymás testvérei az `<egyszerű><összetett><vegyes>` elemek. Összetett az `<összetett>` elem. Egyszerű elem az `<egyszerű>`. Vegyes elemre példa a `<vegyes>` elem.

2.3.2 XML-attribútumok

Egy XML-elemnek tetszőleges számú tulajdonsága lehet, melyeket attribútumoknak nevezünk. Egy attribútum ugyanabban az elemben csak egyszer szerepelhet, értékeik csak egyszerű típusok lehetnek. Egy attribútummal rendelkező elem mindig összetett típusú.

Az attribútumok olyan esetben hasznosak, ha olyan információt hordoznak, melyek nem kapcsolódnak szorosan az elem mondanivalójához. Használatának előnye, hogy logikailag tisztább, átláthatóbb kódot eredményez.

Lássunk egy példát:

```
<elemattrib1="érték" attrib2="érték" attrib3="érték" />
```

2. példa: XML-attribútumok

2.3.3 XML-névterek

Az elem és attribútum nevek egyediek az adott dokumentumon belül, ezeket lokális neveknek hívjuk. Több XML-dokumentum egyesítésénél fennáll a névütközés veszélye, ennek elkerülésére szolgálnak az XML-névterek. Egy XML névtér nem más, mint az XML dokumentumokban elemek és tulajdonságok neveként használt nevek összessége.

Az XML-névtér specifikáció két részből áll, egy rövidebb általában beszédes név, ezt használjuk prefixnek (előtag), és egy hosszabb azonosító, a névtér neve, ami egy URI. A prefix megadása opcionális, elhagyása alapértelmezett névtér megadását teszi lehetővé, ekkor a névtér neve URI helyett lehet üres karakterlánc is.

A dokumentumban felhasznált, adott névtérbe tartozó elem- és attribútum neveket a prefix felhasználásával adhatjuk meg, az alábbi alakban:

```
<prefix:elem>…<elem/>
```

3. példa: Névtér prefix

Egy prefixszel minősített lokális nevet univerzális vagy minősített névnek nevezzük. Ha nem adunk meg prefixet, akkor az alapértelmezett névtérbe fog tartozni az elem.

Névtér-deklarációkat meg lehet adni XML dokumentumokban explicit módon, de szolgáltathatja őket alapértelmezett tulajdonságérték is. Új névteret az `xmlns:prefix` vagy `xmlns` attribútumokkal vehetünk fel, melyek bármely elemhez kapcsolhatók. A névtér hatásköre a tartalmazó elem nyitócímkéjénél kezdődik és tart a megfelelő zárócímkéig. Létezik a „lyuk a hatáskörben" jelenség, azaz érvényben lévő névterek felüldefiniálhatóak. Egy prefixet is deklaráló névtér hatásköre vonatkozik minden olyan elem- és tulajdonságnévre, amelyet az adott prefixszel jelöltünk meg. Alapértelmezett névtér-deklaráció hatásköre vonatkozik minden előtag nélküli elemnévre, azonban nem vonatkozik az előtag nélküli tulajdonságnevekre.

```
<?xml version="1.0"?>
<book xmlns='urn:loc.gov:books'
      xmlns:isbn='urn:ISBN:0-395-36341-6'>
    <title>Cheaper by the Dozen</title>
    <isbn:number>1568491379</isbn:number>
</book>
```

4. példa: Névtér deklarációk

(forrás: http://www.w3.org/TR/xml-names/#scoping-defaulting)

A fenti példában egy alapértelmezett és egy prefixel ellátott névtér deklaráció látható. A `<title>` elem az alapértelmezett, `'urn:ISBN:0-395-36341-6'` névtérbe tartozik, míg az `<isbn:number>` elem az `isbn` prefixszel deklarált `'urn:ISBN:0-395-36341-6'` névtérbe.

2.4 XML-dokumentumok érvényessége, XML-sémák

2.4.1 Helyesen formázott XML-dokumentumok

Egy helyesen formázott XML dokumentumnak (többek között) a következő szabályoknak kell megfelelnie:

- Egyetlen gyökér elem lehet egy dokumentumban. Azonban az XML deklaráció, feldolgozó utasítások és megjegyzések megelőzhetik a gyökér elemet.
- A nem üres elemeket mind nyitó, mind záró címkékkel kell határolni.
- Az üres elemek megjelölhetők üres elem (önlezáró) címkével.
- Minden attribútum érték idézőjelek között van, vagy szimpla(') vagy dupla(") idézőjelek között.

- Az elemek egymásba ágyazhatók, de nem lehetnek átfedők. Mindegyik nem gyökér elemet másik elemnek kell magában foglalnia.
- A dokumentum megfelel a karakterkészlet definíciónak.
- Az elem nevek kisbetű-nagybetű érzékenyek.

2.4.2 Érvényes XML-dokumentumok

Amikor létrehozunk egy új nyelvet az XML segítségével, akkor ezt az XML alkalmazásának hívjuk. Egy konkrét XML-dokumentum egy ilyen nyelvnek a példánya. Az XML-sémák szolgálnak a nyelvek szabványos leírására, segítségével a feldolgozó alkalmazás ellenőrizni tudja, hogy a dokumentum valóban az általa feldolgozni képes nyelvtan példánya-e. Egy XML dokumentum érvényes, ha helyesen formázott és megfelel egy adott sémának.

2.5 XML információs halmazok (XML information set)

Az XML információs halmaz (infoset) egy W3C specifikáció, mely az XML alapjait, az absztrakt adatmodelljének leírását tartalmazza, más szóval az XML-dokumentumok információtartalmának szerkezetét írja le. Egy XML-dokumentumhoz létrehozható az információs halmaza, ha jól formázott és teljesíti a névtér-megszorításokat (nem szükséges érvényesnek lennie).

Egy információs halmaz meghatározza az XML szintaktikai szerkezetének megfelelő információs elemeket (information item). Ezek a következő tizenegy félék lehetnek

1. Dokumentum (The Document Information Item). kötelezően tartalmaznia kell egy információs halmaznak.
2. Elem (Element Information Items).
3. Attribútum (Attribute Information Items
4. Feldolgozási utasítás (Processing Instruction Information Items
5. Nem kiterjesztett egyed (Unexpanded Entity Reference Information Items).
6. Karakter (Character Information Items).
7. Megjegyzés (Comment Information Items).
8. Dokumentum típus deklarációs (The Document Type Declaration Information Item).
9. Nem elemzett egyed (Unparsed Entity Information Items).
10. Jelölés (Notation Information Items).
11. Névtér (Namespace Information Items)

2.6 XML bázis (XML Base)

Minden XML-dokumentumhoz tartozik egy azonosító, ún. URI, melyet a dokumentum bázis (Base) URI-jának nevezünk. Alapesetben ez az az URI, ahol a dokumentum található (például webszerver, fájlrendszer). Az XML lehetőséget ad ennek a bázis URI-nak az explicit módon történő felülírására. Ezt az `xml:base` attribútum használatával tehetjük meg, mely XML-elemekhez kapcsolható. Az attribútum értéke az új bázis URI, melyet az aktuális elemen belül lévő relatív URI-k feloldásához használjuk.

3 RDF

Az RDF (**R**esource **D**escription **F**ramework) egy keretrendszer a Weben történő információábrázolás céljára. Az RDF a W3C (World Wide Web Consortium) ajánlásai közé tartozik, melyet 1999-ben publikáltak. Az RDF olyan absztrakt szintaxissal rendelkezik, mely egy egyszerű gráf alapú adatmodellre épül, valamint szemantikájában a *következmény* (*entailment*) fogalma szigorúan definiált. Ennek alapján az RDF adatokból jól megalapozott információkat vezethetünk le.

Az RDF fejlesztésének motivációi közé tartoztak, hogy metainformációt társíthassunk webes erőforrásokhoz, valamint olyan alkalmazások, szoftver ágensek fejleszthetők, melyek képesek ezen információk automatikus feldolgozására, több különböző alkalmazástól származó adatok kombinálása annak érdekében, hogy ily módon újabb információkhoz juthassunk. Az RDF révén megvalósulhat az, hogy az adatok feldolgozhatók legyenek azon a körön kívül is, amelyben azokat létrehozták.

Tervezési szempontjai a következők:

- Szókészlete URI alapú, korlátlanul bővíthető.
- Az adatmodell legyen egyszerű és független a sorosítási (szerializációs) szintaxistól.
- A formális szemantikája megbízható, egy RDF kifejezés jelentéséből történő következtetés bizonyítható.
- Használjunk XML alapú szerializációs szintaxist, mellyel kódolhatjuk az adatmodellt a gépek közötti adatcsere céljából.
- Bárki tehessen kijelentéseket bármilyen erőforrásról.

3.1 URI

Az RDF segítségével tetszőleges erőforráshoz szabványos módon társíthatunk metainformációt. Az RDF elképzelés szerint erőforrás minden, ami rendelkezik Általános Erőforrás Azonosítóval (**U**niversal **R**esource **I**dentifier), röviden URI-val. Az URI-knak alapvető szerepük van a szemantikus web világában. Az URI-k rövid karaktersorozatok, melyek egyértelműen azonosítanak erőforrásokat. Erőforrások általában a Weben található objektumok, például egy weblap, egy kép, hanganyag vagy video, de erőforrás lehet a Webtől függetlenül is bármi, ami egyedileg azonosítható, azaz URI társítható hozzá, például egy cég vagy személy, vagy egy elvont fogalom például „szerző”. Különböző személyek vagy szervezetek egymástól függetlenül kreálhatnak URI-kat, és ezeket bárminek az azonosítására használhatják.

Speciális URI-k a szélesebb körben ismert URL-ek (**U**niversal **R**esource **L**ocator – Egységes Erőforrás-helymeghatározó), valamint az URN-ek (**U**niversal **R**esource **N**ame - Egységes

Erőforrás-név). Az előbbiek az erőforrásokat az elérési módjukkal azonosítják, míg az utóbbiak perzisztens, azaz mindig rendelkezésre álló erőforrásokat látnak el egyedi névvel. Az elektronikus tartalom azonosítására az interneten egyelőre szinte kizárólag az URL azonosítókat használják, amely minden egyes dokumentum esetében annak lelőhelyét (fizikai helyét) adja meg. Elvileg ez az azonosítási rendszer is lehetővé teszi a dokumentumok hosszú távú azonosítását, ám az elterjedt gyakorlat alapján ezek tartósságára nem lehet számítani, mert igen sokszor előfordul, hogy egy dokumentum lelőhelye megváltozik. Erre a problémára jelentenek megoldást az elérési helytől független egyedi azonosítók, az URN-ek. Ezek az azonosítók nem tartalmaznak információt a dokumentumok tényleges lelőhelyéről, így alapját képezhetik egy hosszú távon is koherens hivatkozásrendszernek. Természetesen a dokumentum eléréséhez annak konkrét lelőhelyére (URL-jére) is szükség van, ám az adott URN-hez tartozó tényleges lelőhelyet elegendő csupán egy központi helyen megadni és frissíteni, ahonnan aztán az URN alapján történő kereséskor ez kiolvasható. Amennyiben a dokumentum gazdája a lelőhely megváltoztatásakor az URN-azonosító, és a lelőhely közti hozzárendelést ezen a központi helyen frissíti, az azonosítón keresztül történő külső és belső hivatkozások koherenciája adminisztrációs munka nélkül is automatikusan megmarad. Tehát az URN-ek szabványosított azonosítók, amelyeket már számos helyen használnak. A jelenlegi szabványok már azt is lehetővé teszik, hogy olyan infrastruktúra épüljön ki hosszútávon, amely révén egy böngésző automatikusan meg tud keresni egy dokumentumot annak URN-je alapján. Az URN-ek szintaxisa:

```
<URN> ::= "urn:" <NID> ":" <NSS>
```

, ahol `<NID>` a névtér azonosító (**N**amespace **Id**entifier), az `<NSS>` egy névtér specifikus karaktersorozat (**N**amespace **S**pecific **S**tring).

Példák URN-ekre:

A "The Last Unicorn" című könyv URN-je:

```
urn:isbn:0451450523
```

A "Pókember (film)" című film URN-je:

```
urn:isan:0000-0000-9E59-0000-O-0000-0000-2
```

Az URI-knak megkülönböztetünk két fajtáját, az abszolút URI-kat, melyek egyértelműen azonosítanak erőforrásokat, valamint relatív URI-kat, melyeknek csak akkor van értelmük, ha rendelkezésre áll egy ún. bázis URI, melynek segítségével feloldjuk, és abszolút azonosítót készítünk belőle.

Az RDF URI hivatkozásokat használ. Egy URI hivatkozás (vagy URIref) nem más, mint egy URI egy opcionális erőforrásrész-azonosítóval (fragment identifier) a végén, melyet egy „#” karakter választ el az URI törzsétől.

A következőkben bemutatok néhány példát URI-kra, illetve URI hivatkozásokra:

```
file://C:/Documents and Settings/Perge Frigyes/Asztal/Diplomamunkám.doc

http://dbtune.org/musicbrainz/resource/track/39282083-62b4-4606-9ad9-
0e6a5c7e0076

mailto: pfrigyes@gmail.com
```

5. példa: URI-k

3.2 Az RDF adatmodell

Mint azt az előzőekben említettem, az RDF célja, hogy metaadatokat társíthassunk erőforrásokhoz. Ennek érdekében a specifikáció definiál egy halmazelméleti alapokon nyugvó adatmodellt, az RDF adatmodellt, melyben a következő halmazokat definiálták:

Erőforrások (Resources): Ezeket már fentebb ismertettem. Röviden: Az összes olyan dolog halmaza, melyeket URI-val azonosíthatunk, amire egy RDF kijelentés vonatkozhat.

Tulajdonságok (Properties): A tulajdonságok erőforrásokhoz kapcsolható jellemzők, melyek maguk is erőforrások, azaz URI-val azonosíthatók. Minden tulajdonságnak van jelentése, meghatározható, mely erőforráshoz kapcsolódhatnak, mi lehet az értékük, valamint milyen viszonyban vannak más tulajdonságokkal.

Literálok (Literals): A literálok egyszerű karaktersorozatok, olyan értékek azonosítására használhatunk, mint pl. számok vagy dátumok. Az azonosítás az értékek lexikális ábrázolása útján történik. Bármit, amit literállal ábrázolhatunk, akár URI hivatkozással is ábrázolhatnánk, de gyakran célszerűbb egy literál használata. A literálok lehetnek típus nélküliek vagy tipizáltak. A tipizált literál szintén egy karakterlánc, amit egy adattípusra mutató URI-vel kombinálunk. Például:

```
"27"^^<http://www.w3.org/2001/XMLSchema#integer>
```

6. példa: Tipizált literál

Kijelentések (Statements): A halmaz elemei hármasok, vagy más néven tripletek, melyek egyszerű kijelentő mondatok. Minden triplet egy kijelentést ábrázol, amely

1. egy alanyból,
2. egy állítmányból, mely viszonyt fejez ki (tulajdonság),

3. és egy tárgyból áll.

Egy RDF triplet azt jelenti ki, hogy az alany és a tárgy által megjelölt dolgok között az állítmány által megfogalmazott viszony áll fenn. Az adatmodell jelentése (szemantikája), hogy a kijelentések halmazába tartozó tripletek igazak.

Példa tripletre az alábbi angol nyelvű mondat:

```
http://www.kispalesaborz.hu has an owner whose value is Kispál és a
Borz
```

7. példa: Egy egyszerű kijelentés

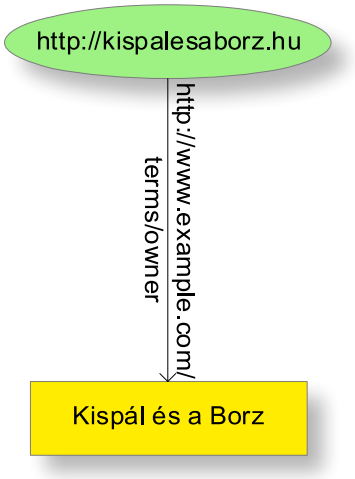
Ez a mondat azt fejezi ki, hogy a `http://www.kispalesaborz.hu` weboldal `tulajdonosa` a `Kispál és a borz`.

3.2.1 Az RDF gráf modellje

Egyszerű kijelentések ábrázolása gráfként

Az RDF tripletek egy halmaza egy címkézett, irányított gráfot határoz meg. Egy RDF gráf csomópontjainak halmazát a gráf tripletjeinek alanyai és tárgyai alkotják, az éleket az állítmányai címkézik. Minden tripletet egy csomópont-él-csomópont kapcsolat reprezentál. Az él mindig az alany csomóponttól a tárgy csomópont felé mutat.

Például:

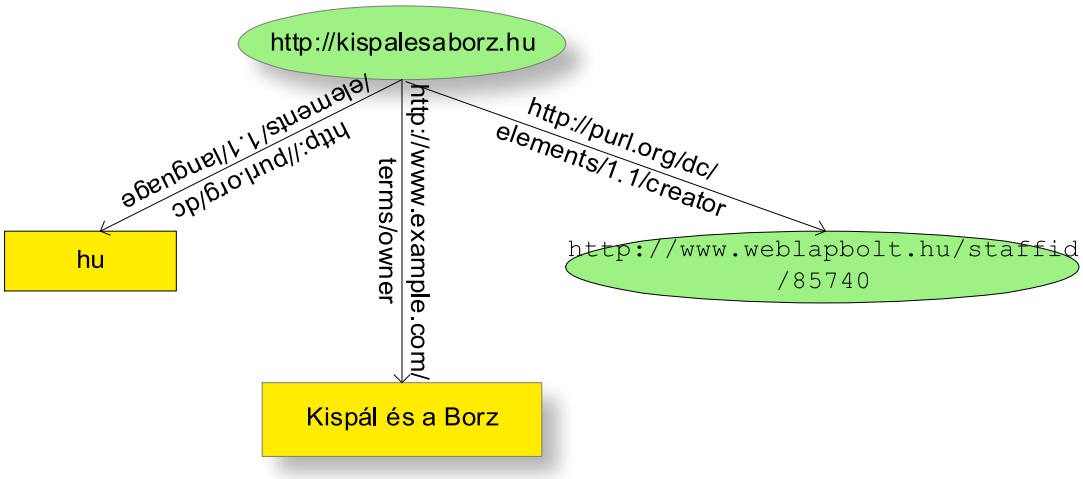


2. ábra: Egy elemi RDF kijelentés ábrázolása

Az RDF gráfok megrajzolásakor azokat a csomópontokat, amelyeket URI-vel azonosítunk, ellipszissel ábrázoljuk, míg az olyan csomópontokat, amelyeket literállal adunk meg, a szögletes dobozok reprezentálják.

Ehhez hasonlóan ábrázolhatunk több kijelentést ugyanarról az erőforrásról.

Például:

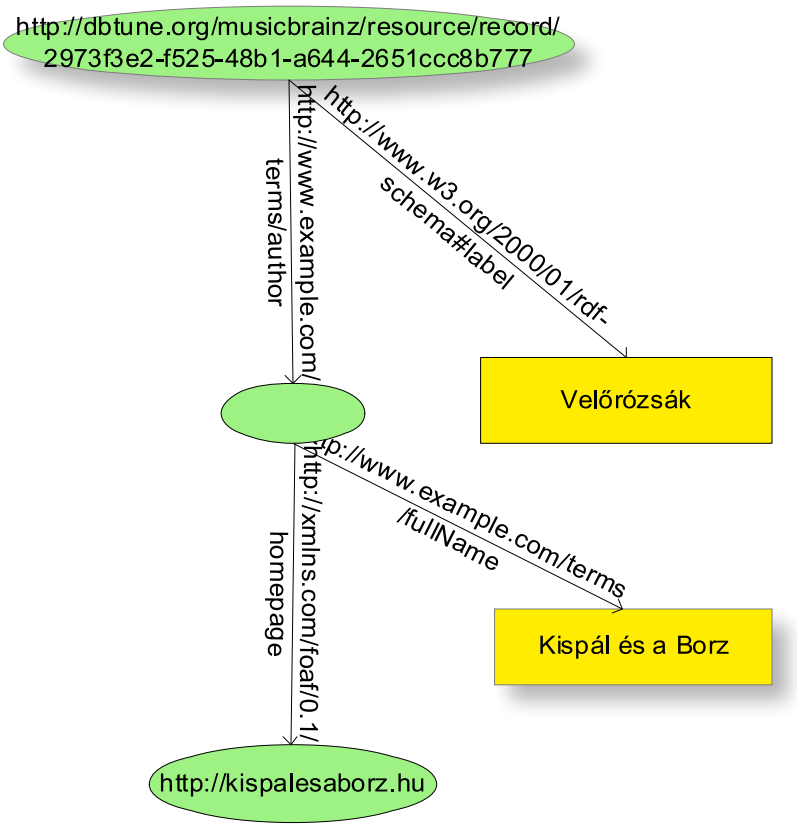


4. ábra: Több kijelentés ugyanarról az erőforrásról

Egy RDF gráf jelentése nem más, mint a benne lévő összes triplet által megfogalmazott kijelentés konjunkciója (logikai ÉS kapcsolata).

Strukturált adatok a gráfban

Mivel a való világ adatai általában nem ilyen egyszerű formában állnak rendelkezésünkre, ezért az RDF-ben lehetőség van strukturált adatok leírására is. Strukturált információt úgy ábrázolunk RDF-ben, hogy az összetett adatot egy önálló erőforrásnak tekintjük, és azután elemi kijelentéseket fogalmazunk meg erről az új erőforrásról. Ezek a gráfban üres csomópontokként jelennek meg, melyek *anonimous resources*, azaz *névtelen erőforrásokat* reprezentálnak. Mivel azonban az ilyen segédfogalmakra általában nem kell az adott gráfon kívülről hivatkozni, ezért általában nincs is szükség globális azonosítókra az elérésükhöz. Az üres csomópontok alkalmazása azt is lehetővé teszi, hogy pontosabban fogalmazhassuk meg állításainkat az olyan erőforrásokról, amelyeknek esetleg nincs URI-jük, de amelyek leírhatók más, olyan erőforrásokhoz való viszonyuk alapján, amelyeknek van URI-jük.



4. ábra: Üres csomópont a gráfban

A példámban egy üres csomópont ábrázolja az album szerzőjét, melyet a teljes neve, illetve a honlapja azonosít.

3.2.2 A tripletek módszere az RDF kijelentések leírására

A tripletek módszere (triples) egy alternatív módja az állítások leírásának. A tripletekkel történő ábrázolás során a gráfban szereplő minden kijelentést egy egyszerű alany-állítmány-tárgy hármassal írunk le, ebben a sorrendben.

Például:

```
<http://kispalesaborz.hu><http://purl.org/dc/elements/1.1/creator><ht
tp://www.weblapbolt.hu/staffid/85740> .

<http:// kispalesaborz.hu><http://www.example.org/terms/creation-
date> "August 16, 1999" .

<http:// kispalesaborz.hu><http://purl.org/dc/elements/1.1/language>
"hu" .
```

8. példa: Tripletes írásmód

A tripletek pontosan ugyanazt az információt reprezentálják, mint a rajzolt gráf.

Az üres csomópontok saját azonosítót kapnak, ún. ürescsomópont-azonosítókat, amelyeket _:name formában írunk. . Az ilyen azonosítók arra szolgálnak, hogy amikor a gráfot tripletek formájában írjuk le, meg tudjuk különböztetni, hogy melyik üres csomópontra hivatkozik egy adott kijelentés.

Most hogy már ismerjük az alapfogalmakat, pontosan megfogalmazhatjuk, hogy egy RDF kijelentés miből állhat. Tehát egy RDF triplet három komponensből áll:

1. az alanyból, ami egy RDF URI hivatkozás, vagy egy üres csomópont
2. az állítmányból, ami egy RDF URI hivatkozás
3. a tárgyból, mely egy RDF URI hivatkozás, egy literál vagy egy üres csomópont.

Összefoglalva, az RDF lényegében egyszerű kijelentések halmazai, melyeket csomópont-él-csomópont diagramokkal ábrázolhatunk.

3.3 RDF/XML

3.3.1 Egy egyszerű kijelentés leírása RDF/XML-ben

Az RDF rendelkezik egy XML szintaxissal, amelyet az RDF gráfok leírására és alkalmazások közötti cseréjére használ, ennek a neve RDF/XML. A szintaxist legkönnyebben egy példán keresztül lehetne bemutatni. Nézzünk egy egyszerű kijelentést RDF/XML-ben, mely szerint a **http://kispalesaborz.hu** weboldal **keletkezési dátuma1999 augusztus 16**.

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.             xmlns:exterms="http://www.example.org/terms/">

4.   <rdf:Description rdf:about="http://kispalesaborz.hu">
5.       <exterms:creation-date>August 16, 1999</exterms:creation-date>
6.   </rdf:Description>

7. </rdf:RDF>
```

9. példa: Egy egyszerű kijelentés RDF/XML alakban

Az 1. sor az, XML deklarációt tartalmazza, mely azt adja meg, hogy a következő tartalom XML-ben, és hogy milyen verziójú XML-ben van ábrázolva.

A 2. sorban szereplő `rdf:RDF` teggel megnyitunk egy elemet, mely a 7.sorban a `</rdf:RDF>` záróteggel ér véget. Ez jelzi, hogy RDF adatot tartalmaz. Ugyanebben és a következő sorban találunk még két névtér deklarációt, melyeket `xmlns` attribútumokként adtunk meg. Az `rdf:` prefixet viselő nevek az RDF szókészlet kifejezései, az `exterms:` prefixet viselő nevek ahhoz a szókészlethez tartoznak melyet a példákban szereplő `http://www.example.org` definiált.

A 4-6. sorok azt az RDF/XML kódot tartalmazzák. A 4. sorban látható `rdf:Description` nyitóteg jelzi az erőforrás leírásának a kezdetét, amelyet szorosan követ annak az erőforrásnak (*alany*) az azonosítása, amelyről a kijelentés szól. Ez az `rdf:about` attribútum segítségével történik, amelynek értéke az erőforrást azonosító URIref. Az 5. sor a tulajdonság-elemet ábrázolja, amit az `exterms:creation-date` minősített név jelöl meg, és amely a kijelentés *állítmányát* azonosítja. Ennek a tulajdonság-elemnek a tartalma a kijelentés *tárgya*, amit az "August 19, 1999" literál ábrázol. A 6. sorban a zárótegje látható.

Tehát az *állítmányok* URI hivatkozásait XML minősített nevekkel jelöltük meg, amelyek egy rövid prefixet tartalmaznak. Az alanyt ábrázoló csomópontok URI hivatkozásait XML attribútum-értékként adtuk meg, és néha ugyanígy adjuk meg a tárgyat ábrázoló csomópont URI hivatkozását is. A literális csomópontok, amelyek mindig tárgyat jelölő csomópontok, a tulajdonság-elemek szövegtartalmaként vagy attribútum-értékeként jelennek meg.

3.3.2 Több kijelentés rövidített megadása

További kijelentéseket hasonló módon, az előző kijelentés után írva adhatunk meg. Az RDF/XML szintaxis biztosít számunkra rövidítési lehetőséget is, hogy a gyakrabban előforduló dolgokat egyszerűbben leírhassuk. Például, amikor egy adott erőforrást egyszerre több tulajdonsággal, illetve értékkel kell jellemeznünk, akkor ezeket tipikusan egyetlen `rdf:Description` elembe ágyazva adjuk meg, amelyik ilyenkor az összes állítmány alanyát reprezentálja.

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.          xmlns:dc="http://purl.org/dc/elements/1.1/"
4.          xmlns:exterms="http://www.example.org/terms/">

5.      <rdf:Description rdf:about="http://kispalesaborz.hu">
6.      <exterms:creation-date>
                August 16, 1999
        </exterms:creation-date>
7.      <dc:language>hu</dc:language>
8.      <dc:creator rdf:resource=
                "http://www.weblapbolt.hu/staffid/85740"/>
9.      </rdf:Description>

10. </rdf:RDF>
```

3.3.3 Üres csomópontok kezelése

Az RDF/XML-ben ábrázolhatunk olyan gráfokat is, amelyekben *üres csomópontok* vannak, ezek olyan csomópontok, amelyekhez nem tartozik URIref. Az RDF/XML-ben egy módszer az olyan gráfok ábrázolására, amelyek üres csomópontokat tartalmaznak, hogy egy ürescsomópont-azonosítót rendelünk minden üres csomóponthoz. Egy ilyen azonosító csak az adott dokumentumon belül képes egy csomópontot azonosítani.

A 9. példa azt mutatja be, hogy az `rdf:nodeID` segítségével hogyan lehet RDF/XML-ben leírni az 5. ábrán szereplő gráfot:

```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.             xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
4.             xmlns:exterms="http://example.org/stuff/1.0/
5.             xmlns:foaf="http://xmlns.com/foaf/0.1">

6.     <rdf:Description rdf:about=
7.             "http://www.w3.org/TR/rdf-syntax-grammar">
8.       <rdfs:label>Velőrózsákdc:title>
9.       <exterms:author rdf:nodeID="abc"/>
10.     </rdf:Description>

11.     <rdf:Description rdf:nodeID="abc">
12.        <exterms:fullName>Lovasi András</exterms:fullName>
13.        <foaf:homepage rdf:resource="http://kispalesaborz.hu"/>
14.    </rdf:Description>

15. </rdf:RDF>
```

Az üres csomópontra az RDF-ben az `rdf:nodeID` attribútum segítségével hivatkozunk, amelynek az értéke egy ürescsomópont-azonosító, a fenti példában ez az `"abc"` azonosító. Az olyan kijelentéseket, amelyeknek az alanya egy üres csomópont, az RDF/XML-ben egy olyan `rdf:Description` elemmel is leírhatjuk, amelyben az `rdf:about` helyett egy `rdf:nodeID` attribútumot használ, ahogy a 11. sorban látható. Hasonlóan használható az `rdf:resource` attribútum helyett egy `rdf:nodeID` attribútum, a tárgy megadásánál (9. sor).

A módszer előnye, hogy ennek segítségével egy RDF/XML dokumentum több helyéről is hivatkozhatunk ugyanarra az üres csomópontra.

3.3.4 Tipizált literálok

Az előző példákban megjelent típus nélküli literálok helyett használhattam volna tipizált literálokat is tulajdonságok értékeként, melyek jóval több információt nyújtanak az értékük korrekt meghatározásához. Egy tipizált literált úgy ábrázolunk, hogy a tulajdonság-elemhez, mely közrefogja a literált, egy `rdf:datatype` attribútumot adunk, amelynek értéke egy, a literál adattípusát azonosító URIref.

A 7. példa tipizált literállal:

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.             xmlns:exterms="http://www.example.org/terms/">

4.   <rdf:Description rdf:about="http://kispalesaborz.hu">
5.     <exterms:creation-date rdf:datatype=
         "http://www.w3.org/2001/XMLSchema#date">1999-08-16
</exterms:creation-date>
6.   </rdf:Description>

7. </rdf:RDF>
```

12. példa: Tipizált literált használó RDF/XML leírás

3.3.5 Az rdf:ID attribútum, XML bázis opció

Az eddigiekben olyan erőforrásokat írtunk le, melyeket azonosított legalább egy URIref. Sok esetben azonban olyan erőforrásokról szeretnénk RDF leírást készíteni, melyeknek vagy még nincs URIref-jük, vagy pedig nem ismerjük mi az. Az ilyen esetekben az `rdf:about` attribútum helyett az `rdf:ID` használhatjuk, az RDF-kijelentések alanyainak azonosításához. Használatának jelentősége abban áll, hogy nagyon hasznos lehet például akkor, ha egy dokumentumon belül különböző, de valamilyen szempontból összetartozó dolgokról szeretnénk leírást adni, például egy együttes albumairól. Az rdf:ID speciális tulajdonsága, hogy egy adott bázis URI hatáskörében (ez általában a dokumentum bázis URI-ja), egy vele megnevezett azonosító csak egyszer fordul elő. Ezt az RDF-elemzőknek kötelezően ellenőrizniük kell.

Az alanyokat azonosító abszolút URIref-eket úgy képezzük, hogy vesszük az aktuális bázis URI-t, majd hozzáírjuk a # karaktert, majd az `rdf:ID` attribútum értékét.

Az RDF/XML támogatja az XML bázis opciót, mely lehetővé teszi, hogy az XML dokumentum explicit módon specifikáljon egy olyan bázis-URI-t, amelyik független a dokumentum saját bázis-URI-jától. Ekkor az alany abszolút URIref-jének előállításánál ezt az új bázis URI-t használjuk a dokumentumé helyett. Ez akkor fontos, ha például egy cég egy mirror site-on is szeretné publikálni az RDF dokumentumait, és másolatot készít róluk, akkor a másolatnak más lesz a bázis URI-ja, mint az eredeti dokumentumnak, és a másolatban szereplő erőforrások abszolút URIref-jei már ez alapján állnának elő, tehát nem a kívánt

erőforrást azonosítanák. A bázis URI explicit megadása, függetleníti a relatív URIref-ek feloldását a dokumentum publikálásának helyétől, saját bázis URI-jától.

```
1. <?xml version="1.0"?>
3. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.      xmlns:dc="http://www.purl.org/dc/elements/1.1/"
5.      xml:base="http://dbtune.org/musicbrainz/resource/record/">

6.      <rdf:Description rdf:ID="2973f3e2-f525-48b1-a644-2651ccc8b777">
7.              <dc:date> 2000-00-00 </dc:date>
8.              <dc:language rdf:resource=
9.                "http://dbtune.org/musicbrainz/resource/language/hun"/>
10.             <dc:title>Velőrózsák</dc:title>
11.     </rdf:Description>

  ...további albumok leírásai...

12.  </rdf:RDF>
```

bázishoz viszonyítva értendő, mindegy, hogy mi az aktuális dokumentum saját bázis-URI-ja. Ennek eredményeképpen a 6. sorban lévő az albumot azonosító `rdf:ID="2973f3e2-f525-48b1-a644-2651ccc8b777"` relatív hivatkozásból, a következő abszolút hivatkozást kapjuk: `http://dbtune.org/musicbrainz/resource/record/#2973f3e2-f525-48b1-a644-2651ccc8b777`.

3.3.6 Az rdf:type attribútum

A dolgok különböző *fajtákba* vagy *kategóriákba* történő besorolása hasonló, a programnyelvekben használt objektumok különböző *típusokba* vagy *osztályokba* sorolásához. Az RDF támogatja ezt a fogalmat egy előre definiált tulajdonság, az `rdf:type` bevezetésével. Amikor egy RDF erőforrást valamely tulajdonságával leírunk, akkor ennek a tulajdonságnak az értékét egy olyan erőforrásnak tekintjük, mely a dolgok egy osztályát képviseli; ennek a tulajdonságnak az alanyát, pedig a saját osztálya egyik előfordulásának (*instance*), vagy más néven egyedének (*individual*) tekintjük.

```
1. <?xml version="1.0"?>
3. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.      xmlns:dc="http://www.purl.org/dc/elements/1.1/"
5.      xml:base="http://dbtune.org/musicbrainz/resource/record/">

6.      <rdf:Description rdf:ID="2973f3e2-f525-48b1-a644-2651ccc8b777">
7.              <rdf:type rdf:resource=
                  "http://purl.org/ontology/mo/Record"/>
8.              <dc:date> 2000-00-00 </dc:date>
9.              <dc:language rdf:resource=
10.               "http://dbtune.org/musicbrainz/resource/language/hun"/>
11.             <dc:title>Velőrózsák</dc:title>
12.     </rdf:Description>

  ...további albumok leírásai...

13.  </rdf:RDF>
```

Az ilyen erőforrásokat a gráfban *tipizált csomópontoknak*, az RDF/XML leírásban pedig *tipizált csomópont-elemeknek* nevezzük.

Az RDF/XML megenged egy speciális rövidítési lehetőséget is:

```
1. <?xml version="1.0"?>
3. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.      xmlns:dc="http://www.purl.org/dc/elements/1.1/"
5.      xmlns:mo="http://purl.org/ontology/mo/"
6.      xml:base="http://dbtune.org/musicbrainz/resource/record/">

7.      <mo:Record rdf:ID="2973f3e2-f525-48b1-a644-2651ccc8b777">
8.              <dc:date> 2000-00-00 </dc:date>
9.              <dc:language rdf:resource=
10.               "http://dbtune.org/musicbrainz/resource/language/hun"/>
11.             <dc:title>Velőrózsák</dc:title>
12.     </mo:Record>

  ...további albumok leírásai...

12.  </rdf:RDF>
```

3.4 Az RDF további lehetőségei

Az RDF számos további lehetőséget is nyújt számunkra, mint például a beépített típusok és tulajdonságok az erőforrások és RDF kijelentések csoportos ábrázolására, illetve az XML kódrészek tulajdonságok értékeként ábrázolása. Ezeket veszem a következőkben sorra.

3.4.1 Konténerek

Az RDF rendelkezik több beépített típussal és tulajdonsággal, amelyeket arra használhatunk, hogy dolgokból csoportokat képezzünk, és így írjuk le őket. Ilyen eset például, ha egy együttes zeneszámait csoportként kell kezelnünk és jellemeznünk.

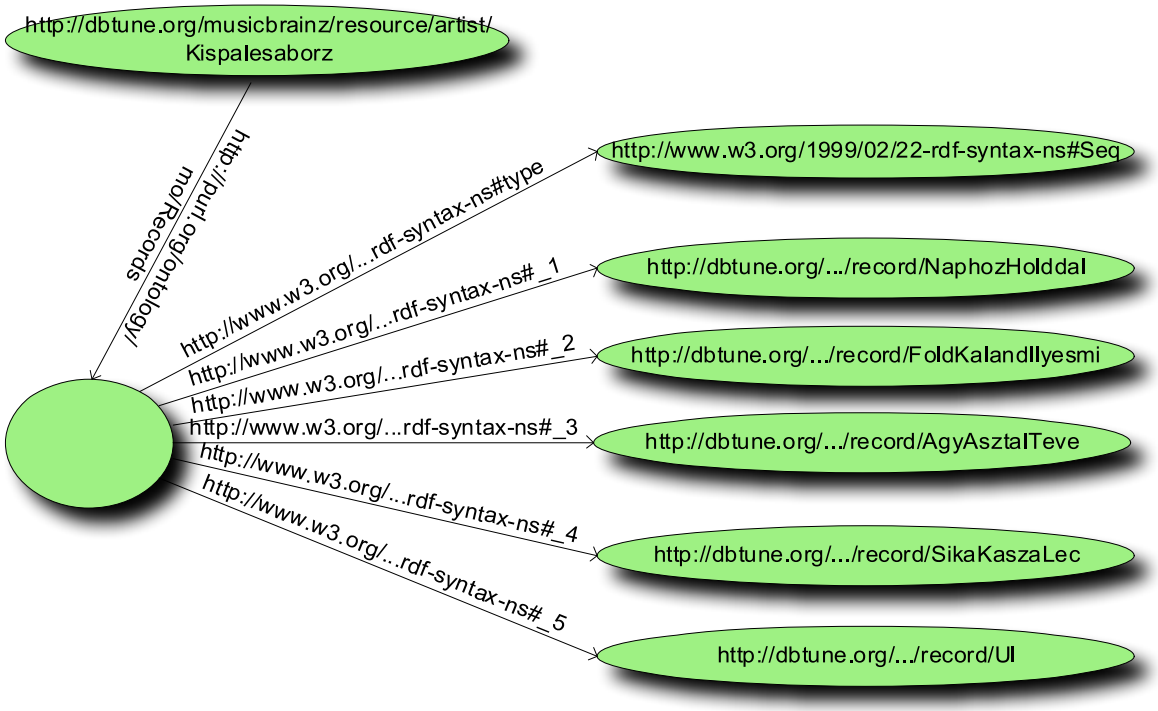
Az RDF-nek van egy *konténer szókészlete*, mely három előre definiált típust tartalmaz, és az ezekhez kapcsolódó tulajdonságokat. A *konténer* egy olyan erőforrás, mely több dolgot tartalmaz, ezeket nevezzük *tagok*nak. A konténer tagjai erőforrások, üres csomópontok és literálok lehetnek.

Három féle konténer típusú erőforrás van:

Zsák (rdf:Bag): A *zsák* egy `rdf:Bag` típusú erőforrás, olyan erőforrások vagy literálok csoportját ábrázolja, amelyben duplikált tagok is előfordulhatnak, és amelyben **nincs** jelentősége a tagok sorrendjének.

Sorozat vagy szekvencia (rdf:Seq): A *sorozat vagy szekvencia* egy `rdf:Seq` típusú erőforrás, olyan erőforrások vagy literálok csoportját ábrázolja, amelyben duplikált tagok is előfordulhatnak, és amelyben jelentősége **van** a tagok sorrendjének.

Alternatíva-csoport (rdf:Alt): Az *alternatíva-csoport* egy `rdf:Alt` típusú erőforrás, olyan erőforrások vagy literálok csoportját ábrázolja, amelyben a tagok egymás alternatívái.



5. ábra: Egy egyszerű Seq konténer leírása gráffal

A `http://purl.org/ontology/mo/Record` tulajdonság értéke az ábrán egy `Seq` konténer formájában van leírva, a sorrend itt fontos, mivel az albumok a megjelenési dátumuk sorrendjében vannak felírva. Ugyanez a példa RDF/XML-ben megadva:

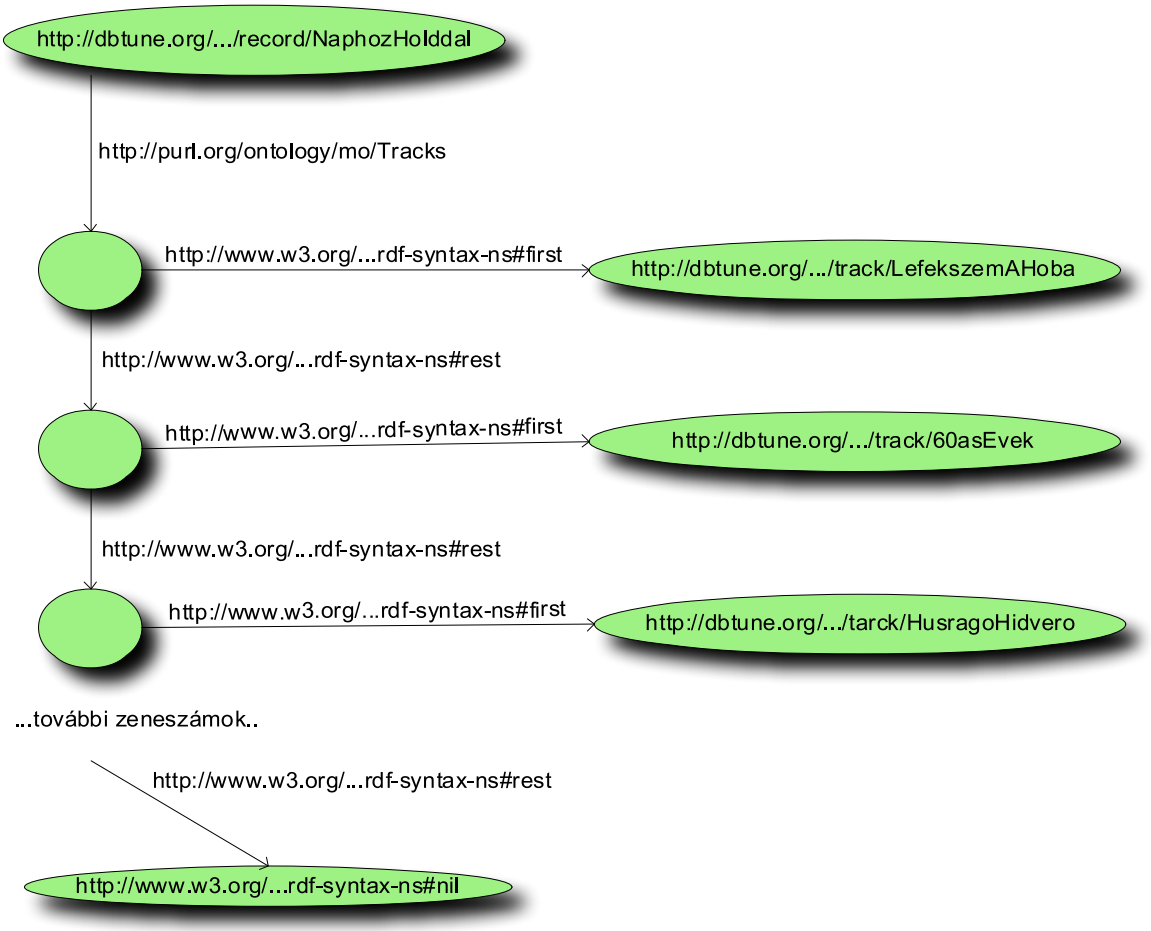
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:mo="http://purl.org/ontology/mo/">

   <rdf:Description rdf:about=
        "http://dbtune.org/musicbrainz/resource/artist/Kispalesaborz">
      <mo:Records>
         <rdf:Seq>
            <rdf:li rdf:resource="http://.../record/NaphozHolddal"/>
            <rdf:li rdf:resource="http://.../record/FoldKalandIlyesmi"/>
            <rdf:li rdf:resource="http://.../record/AgyAsztalTeve"/>
            <rdf:li rdf:resource="http://.../record/SikaKaszaLec"/>
            <rdf:li rdf:resource="http://.../record/Ul"/>
         </rdf:Seq>
      </mo:Records >
   </rdf:Description>
</rdf:RDF>
```

Ahhoz, hogy leírjunk egy erőforrást, mint e konténertípusok egyikét, az erőforrásnak egy `rdf:type` tulajdonságot adunk, amelynek értéke az `rdf:Bag`, `rdf:Seq`, vagy `rdf:Alt` előre definiált erőforrás valamelyike lesz, attól függően, hogy éppen melyikre van szükségünk. A tagságtulajdonságok nevének formátuma `rdf:_n`, ahol *n* egy nem nulla értékű decimális egész szám, ezek szolgálnak a konténer tagjainak leírására.

3.4.2 Kollekciók

A konténerek hasznos eszközök tulajdonság értékek csoportjainak leírására, azonban van egy hiányosságuk mégpedig, hogy nincs lehetőség a lezárásukra, nem mondhatjuk azt, hogy „a csoportnak ez az összes eleme”. Egy konténer csak azt jelenti, hogy néhány erőforrás a tagja, de azt nem fejezi ki, hogy ne lennének más tagjai, valahol máshol a világon ne lenne egy olyan gráf, mely további elemeket írna le. Hogy ezt a „hiányosságot” pótolja, az RDF rendelkezésünkre bocsát egy újabb szókészletet melynek neve *kollekció szókészlet*.

Az RDF kollekciók segítségével leírhatunk olyan csoportokat is, amelyekről biztosan tudható, hogy csakis a leírt tagokat tartalmazzák. Egy RDF kollekció a dolgok egy csoportja, amelyet egy lista-struktúra formájában ábrázolunk az RDF gráfban. A kollekció szókészlet előre definiált tulajdonságból, az `rdf:List` típusból, az `rdf:first` ("első eleme") és `rdf:rest` ("maradék része"), valamint a szintén előre definiált `rdf:nil` ("üres lista") erőforrásból áll, melyek segítségével konstruálhatjuk a lista-szerkezetet. Kollekciót használhatunk például egy zenei albumon található zeneszámok leírásához, ahogy azt a következő ábrán is láthatjuk.



6. ábra: Egy zenei album számainak leírása RDF kollekcióval

Ebben a gráfban a kollekció minden tagja, egy `rdf:first` tulajdonság tárgya, az alanya pedig egy erőforrás (példámban ez üres csomópont), mely a listát képviseli. Ezt a lista típusú erőforrást egy `rdf:rest` tulajdonsággal kapcsoljuk a lista maradék részéhez. A lista végét egy `rdf:nil` értékkel rendelkező `rdf:rest` tulajdonság jelzi, ahol az `rdf:nil` egy `rdf:List` típusú üres listát reprezentál. A fenti gráf RDF/XML alakban:

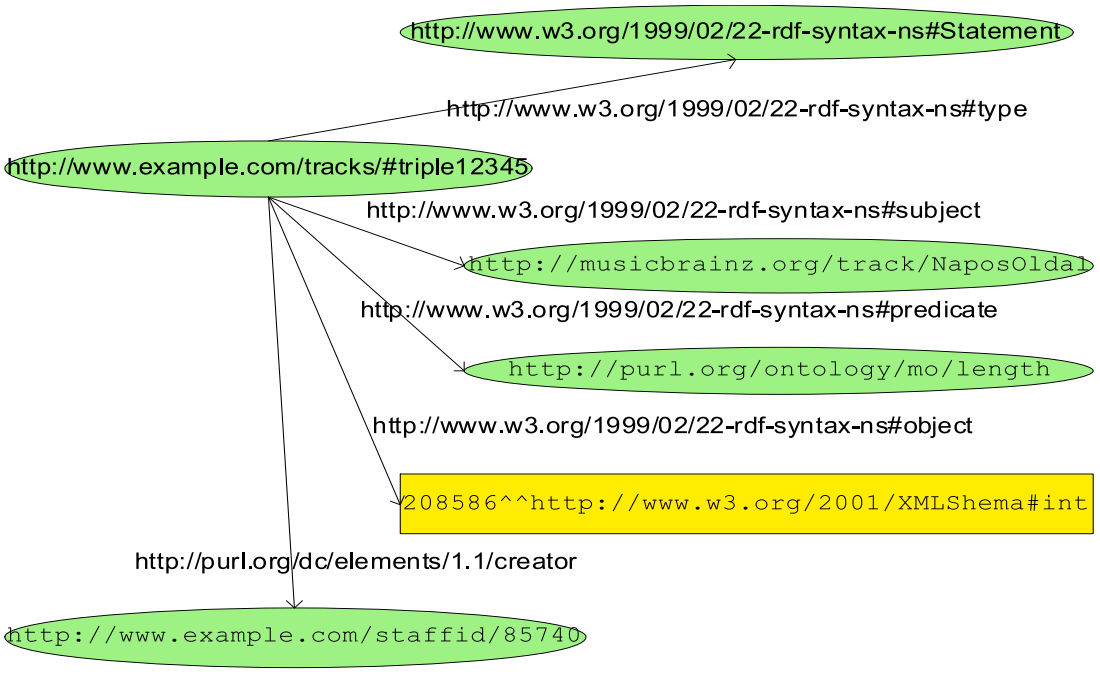
```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://purl.org/ontology/mo">

        <rdf:Description rdf:about=
                "http://dbtune.org/.../record/NaphozHolddal">
          <mo:Tracks rdf:parseType="Collection">
                <rdf:Description rdf:about=
                        "http://dbtune.org/.../track/LefekszemAHoba"/>
                <rdf:Description rdf:about=
                        "http://dbtune.org/.../track/60asEvek"/>
                <rdf:Description rdf:about=
                        "http://dbtune.org/.../track/HusragoHidvero"/>
                …további zeneszámok…
      </mo:Tracks>
   </rdf:Description>
</rdf:RDF>
```

`rdf:parseType="Collection"` attribútuma, és amelynek a tartalma olyan beágyazott elemek egy csoportja, amelyek a kollekció tagjait ábrázolják. Az RDF/XML annak jelzésére használja az `rdf:parseType` attribútumot, hogy az adott elem tartalmát valamilyen speciális módon kell értelmezni. Esetünkben az `rdf:parseType="Collection"` attribútum azt jelzi, hogy a beágyazott elemeket a kollekciónak megfelelő listaszerkezetet ábrázoló gráf előállítására kell használni.

3.4.3 Tárgyiasítás, avagy kijelentések kijelentésekről

Néha szükség van arra, hogy RDF kijelentésekről fogalmazzunk meg állításokat. Ilyen eset például, ha információt kell rögzíteni arról, hogy mikor készítették az adott kijelentéseket, vagy ki készítette őket. Hogy magukról az RDF kijelentésekről is lehet kijelentéseket tenni, az RDF-ben rendelkezésünkre áll egy beépített szókészlet. Azt az eljárást, amelynek során ezzel a szókészlettel leírunk egy kijelentést, *tárgyiasításnak* (*reifikáció*) hívjuk, e művelet eredményét pedig az adott kijelentés *tárgyiasulásának* nevezzük. Az RDF tárgyiasító szókészlete négy elemből,

- az `rdf:Statement` típusból,
- `rdf:subject`
- `rdf:predicate`
- és `rdf:object` tulajdonságokból áll.



7. ábra: Egy kijelentés tárgyiasítása gráf alakban

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:mo="http://purl.org/ontology/mo/"
            xml:base="http://www.example.com/tracks">/>

  <rdf:Description
              rdf:about="http://musicbrainz.org/track/NaposOldal">
       <mo:length rdf:datatype="&xsd;int>
              208586
       </mo:length>
  </rdf:Description>

  <rdf:Statement rdf:about="#triple12345">
       <rdf:subject
         rdf:resource=" http://musicbrainz.org/track/NaposOldal "/>
       <rdf:predicate
         rdf:resource=" http://purl.org/ontology/mo/length"/>
       <rdf:object rdf:datatype="&xsd;int">208586</rdf:object>

       <dc:creator
         rdf:resource="http://www.example.com/staffid/85740"/>
 </rdf:Statement>

</rdf:RDF>
```

Egy a
htt gy
tárg t,
rdf a

tárgyiasított állítást felhasználjuk arra, hogy kijelentsük az állítás létrehozója (creator) a `http://www.example.com/staffid/85740` URI-val azonosított munkatárs.

3.5 RDF sémák

Az RDF felhasználói közösségeknek szükségük van arra is, hogy előre definiálhassák az egyes szakterületek fogalmait tartalmazó szókészleteiket, amelyeket majd a későbbi RDF kijelentéseikben kívánnak alkalmazni. Ezekben a szókészletekben definiálhatják az általuk leírni kívánt erőforrások specifikus *fajtáit* vagy *osztályait*, valamint azokat a specifikus *tulajdonságokat*, amelyekkel majd ezeknek az osztályoknak az *egyedeit* kívánják jellemezni. Gondoljunk például arra, hogy egy lemezkiadó cég, a kiadott lemezeik RDF-ben történő publikálásához szeretne olyan osztályokat definiálni, mint *zenei előadó*, *zenei album*, *zeneszám*, vagy a *szerzője* tulajdonságot. Az RDF maga erre nem nyújt lehetőséget, azonban létezik az RDF nyelvnek egy kiterjesztése, melyet az RDF Szókészlet Leíró Nyelvnek, vagy röviden RDF Sémának nevezünk. Az RDF Séma segítségével az osztályokat és tulajdonságokat, egy RDF szókészlet elemeiként definiálhatjuk.

Az RDF Séma természetesen nem tartalmazza az alkalmazás-specifikus osztályokat, ehelyett olyan eszközöket és módszereket tartalmaz, amelyek szükségesek az alkalmazás-specifikus osztályok és tulajdonságok definiálásához, valamint annak jelzéséhez, hogy mely osztályokat és tulajdonságokat kívánunk együtt használni, azaz egy „típus-rendszert” nyújt számunkra.

Az RDF Séma típus-rendszere néhány szempontból hasonlít, és néhány szempontból jelentősen eltér az objektumorientált programozási nyelvek típus-rendszerétől. Lehetőségünk van az erőforrásokat egy vagy több osztály egyedeként definiálni, az osztályokat hierarchiákba szervezhetjük. Különbség azonban, hogy a programozási nyelvekkel ellentétben nem hozunk létre új osztályokat, mindössze egy már meglévő erőforrásról jelenthetjük ki, hogy az egy osztály.

Az RDF Séma eszköztára maga is egy RDF szókészlet, a szókészletben szereplő erőforrások URI hivatkozásai mind a `http://www.w3.org/2000/01/rdf-schema#` prefixszel érhetők el.

3.5.1 Osztályok

Az osztályokat az `rdfs:Class` („Osztály”) és az `rdfs:Resource` („Erőforrás”) nevű RDF Séma erőforrások, valamint az `rdf:type` („Típus”) és az `rdfs:subClassOf` („Alosztálya”) nevű tulajdonságok segítségével írhatjuk le. Azokat az erőforrásokat, amelyek valamilyen osztályhoz tartoznak, az osztály *példányainak* (instance) vagy *egyedeinek* (individual) nevezzük. Az RDF Sémában osztálynak tekintünk minden olyan erőforrást, amelyiknek a leírásában szerepel egy olyan `rdf:type` tulajdonság, amelynek az értéke az Osztály nevű erőforrás (`rdfs:Class`). Az `rdf:type` tulajdonságot annak kijelentésére használjuk, hogy a vele jellemzett erőforrás valamely osztálynak az egyede. Egy erőforrás több osztálynak is lehet az egyede.

Ha ki szeretnénk jelenteni egy valamiről, hogy az egy osztály a következőket kell tennünk:

- Hozzá kell rendelnünk egy azonosítót, azaz URI-t,
- majd kijelentjük az így azonosított erőforrásról, hogy a típusa Osztály. Ezt úgy tehetjük meg, ha az `rdf:type` tulajdonságnak az `rdfs:Class` értéket adjuk.

Magának az `rdfs:Class` erőforrásnak a típusa szintén `rdfs:Class`.

Osztályhierarchiák

Egy A osztályról az `rdfs:subClassOf` tulajdonság segítségével jelenthetjük ki, hogy az *alosztálya* egy B osztálynak. Ez azt jelenti, hogy A minden példánya, példánya B-nek is. Egy osztálynak tetszőleges számú szülőosztálya és tetszőleges számú alosztály lehet. Az `rdfs:subClassOf` tulajdonság *tranzitív*, azaz ha A alosztálya B-nek, B alosztálya C-nek, akkor A alosztálya C-nek is. Az RDF Séma minden osztályt az Erőforrás (`rdfs:Resource`) osztály alosztályaként értelmez.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base="http://purl.org/ontology/mo/">

<rdf:Description rdf:ID="MusicArtist">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="MusicGroup">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MusicArtist"/>
</rdf:Description>

<rdfs:Class rdf:ID="Track"/>

<rdf:RDF/>
```

MusicGroup osztály a MusicArtist osztály specializációja, azaz alosztálya. Az utolsó előtti sorban látható Track osztály, a „tipizált csomópont” rövidítést használva van leírva. A továbbiakban is ezt a rövidítést használom, ahol lehet.

3.5.2 Tulajdonságok

Az RDF Sémában a tulajdonságokat a `rdf:Property` („Tulajdonság”) nevű RDF osztály , valamint az `rdfs:domain` („érvényességi kör”) az `rdfs:range` („értéktartomány”) és az `rdfs:subPropertyOf` („altulajdonsága”) nevű RDF Séma tulajdonságok, vagy más szóval tulajdonságkorlátozások használatával írjuk le.

Ha ki szeretnénk jelenteni egy valamiről, hogy az egy tulajdonság a következőket kell tennünk:

- Hozzá kell rendelnünk egy azonosítót, azaz URI-t,
- majd kijelentjük az így azonosított erőforrásról, hogy a típusa Tulajdonság. Ezt úgy tehetjük meg, ha az `rdf:type` tulajdonságnak az `rdfs:Property` értéket adjuk.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base="http://xmlns.com/foaf/0.1">

<rdfs:Property rdf:ID="maker"/>

<rdfs:Property rdf:ID="name"/>

<rdf:RDF/>
```

és a hatókörük alapértelmezésben globális, szükség esetén azonban meghatározott osztály(ok)ra korlátozhatjuk. Az RDF Séma egy olyan szókészletet is tartalmaz, amellyel leírhatjuk, hogy mely tulajdonságokat, mely osztályokkal összefüggésben kívánunk használni az RDF adatok leírásakor. Az ilyen jellegű információkat tulajdonságkorlátozásoknak nevezzük, melyeket az `rdfs:domain` („érvényességi kör”) az `rdfs:range` („értéktartomány”) tulajdonságokkal adhatunk meg.

Értéktartomány*:* Az `rdfs:range` tulajdonságot annak jelzésére használjuk, hogy egy adott tulajdonság értéke csak a megjelölt osztály valamelyik egyede lehet. Ezt kell használnunk, ha például azt akarjuk kifejezni, hogy a „szerzője” (`foaf:maker`) tulajdonság csak a „zenei előadók” (`mo:MusicArtist`) osztály egyedei közül kerülhet ki. Egy tulajdonsághoz, hozzárendelhetünk zéró, egy, vagy több ilyen tulajdonságkorlátozást. Ha nem adunk meg egyet sem, akkor ez azt jelenti, hogy nem kívánjuk korlátozni a tulajdonság lehetséges értékeit. Ha megadunk egyet, akkor ezzel azt deklaráljuk, hogy a tulajdonság lehetséges értékei a megadott osztály egyedei. Ha több értéktartományt adunk meg, akkor ezzel azt fejezzük ki, hogy a tulajdonság értéke lehet bármi, ami minden megadott osztálynak egyede. (Megjegyzés: Egy erőforrás több osztálynak is lehet az egyede). Az `rdfs:range` értéke nemcsak valamilyen felhasználó által definiált osztály lehet, hanem egy XML Séma adattípus is, mint pl. `xsd:integer`.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base="http://purl.org/ontology/mo/"

<rdfs:Property rdf:ID="Author">
   <rdfs:range rdf:resource="MusicArtist"/>
<rdfs:Property/>

<rdf:RDF/>
```

21. példa: Értéktartomány tulajdonságkorlátozás leírása

Érvényességi kör: Az `rdfs:domain` tulajdonságot annak jelzésére használjuk, hogy egy adott tulajdonság a megjelölt osztályra vonatkozik (vagyis, csak arra érvényes). Ezt kell használnunk, ha például azt akarjuk kifejezni, hogy a „szerzője” (`foaf:maker`) tulajdonság csak a „zeneszámok” (`mo:Track`) osztály egyedeire legyen érvényes. Egy tulajdonsághoz, hozzárendelhetünk zéró, egy, vagy több ilyen tulajdonságkorlátozást. Ha nem adunk meg egyet sem, akkor ez azt jelenti, hogy nem kívánjuk korlátozni a tulajdonság érvényességi körét. Ha megadunk egyet, akkor ezzel azt deklaráljuk, hogy a tulajdonság csak a megadott osztály egyedire alkalmazható. Ha több korlátozást adunk meg, akkor ezzel azt fejezzük ki, hogy bármely erőforrás, amelynek lehet ilyen tulajdonsága, minden megadott osztálynak egyede kell legyen.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base="http://purl.org/ontology/mo/">

<rdfs:Property rdf:ID="author">
   <rdfs:domain rdf:resource="Track"/>
   <rdfs:range rdf:resource="MusicArtist"/>
<rdfs:Property/>

<rdf:RDF/>
```

22. példa: Érvényességi kör tulajdonságkorlátozás leírása

Tulajdonsághierarchiák

Az RDF Séma az osztályok mellett a tulajdonságok esetében is lehetővé teszi a specializációt. A specializációs viszonyt két tulajdonság között az előre definiált `rdfs:subPropertyOf` (altulajdonsága) RDF Séma tulajdonsággal írjuk le. Egy tulajdonság zéró, egy, vagy több tulajdonságnak is altulajdonsága lehet. Minden `rdfs:range` és `rdfs:domain` tulajdonságkorlátozás, amelyet valamely tulajdonságra megadunk, érvényes annak összes altulajdonságára is. Az `rdfs:subPropertyOf` tulajdonság az `rdfs:subClassOf` tulajdonsághoz hasonlóan *tranzitív*, azaz ha A altulajdonsága B-nek, B altulajdonsága C-nek, akkor A altulajdonsága C-nek is.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xml:base=" http://purl.org/ontology/mo/">

<rdfs:Property rdf:ID="Author">
   <rdfs:domain rdf:resource="Track"/>
   <rdfs:range rdf:resource="MusicArtist"/>
<rdfs:Property/>

<rdfs:Property rdf:ID="LyricsAuthor">
   <rdfs:domain rdf:resource="Track"/>
<rdfs:Property/>

<rdf:RDF/>
```

25. példa: Altulajdonságok leírása

A „LyricsAuthor” altulajdonsága az „Author” tulajdonságnak, örökli a tulajdonság-korlátozásokat, ezért nem szükséges azokat újra megadni.

3.5.3 További sémainformációk

Az RDF Séma rendelkezik több más, beépített tulajdonsággal, amelyek dokumentálják, vagy kommentálják az RDF Sémánkat.

- Az `rdfs:comment` tulajdonsággal ember által olvasható megjegyzést adhatunk az erőforráshoz.
- Az `rdfs:label` (címke) tulajdonsággal egy erőforrás nevének ember által olvashatóbb változatát adhatjuk meg.
- Az `rdfs:seeAlso` (lásd még) tulajdonság utalást jelent egy másik erőforrásra, mely további információt nyújthat az alanyként megadott erőforrásról.

- Az `rdfs:isDefinedBy` tulajdonság `rdfs:seeAlso` tulajdonság altulajdonsága, és egy olyan erőforrás megjelölésére használható, amely valamilyen formában definiálja az alanyként megadott erőforrást.

Az RDF Séma biztosítja az alapvető eszközöket az RDF szókészletek leírására. Az RDF Séma továbbfejlesztésével, vagy más olyan nyelvek alkalmazásával, amelyek az RDF-en alapulnak, további gazdagobb sémaleíró lehetőségekhez jutunk. Ilyen lehetőségek, melyeket az RDF Séma nem tartalmaz, például:

- Kardinalitáskorlátozások specifikálása a tulajdonságokra. A kardinalitás egy olyan tulajdonságkorlátozás, amely megadja, hogy egy adott a tulajdonságnak hány különböző értéke lehet.
- Annak specifikálása, hogy egy adott tulajdonság egy adott osztály valamelyik egyedének egyedi azonosítója (vagy kulcsa).
- Annak specifikálása, hogy két olyan osztály, amelyik két különböző URI hivatkozással érhető el, ténylegesen ugyanazt az osztályt reprezentálja.
- Annak specifikálása, hogy két olyan egyed, amelyik két különböző URI hivatkozással érhető el, ténylegesen ugyanazt az egyedet reprezentálja.
- Új osztályok leírása egy vagy több osztály halmazuniójaként vagy metszeteként, továbbá annak kijelentése, hogy két osztály diszjunkt.

Ezeket a képességeket, sok más hasznos képességgel együtt, próbálják megvalósítani az ontológianyelvek, köztük az OWL is.

4 OWL

A következő elem, ami szükséges a szemantikus webhez, egy ontológianyelv, amellyel formálisan le tudjuk írni a webes dokumentumokban használt terminológia jelentését. Ha elvárjuk a gépeinktől, hogy képesek legyenek hasznos következtetéseket levonni e dokumentumok tartalmából, akkor a leíró nyelv szemantikájának meg kell haladnia az RDF Séma alapvető szemantikáját. Az OWL (**W**eb **O**ntology **L**anguage) Web Ontológia Nyelv egy Web ontológiák tervezésére és előállítására szolgáló nyelv, melyet 2004-es bejelentése óta a W3C ajánlásai között tartanak számon. Az OWL nyelv alapjait a DAML+OIL nyelv képezte.

4.1 Web ontológia

Az *ontológia* kifejezést a filozófiából vették kölcsön, és azt a tudományt jelöli meg, mely a világ entitásainak típusait, és ezek egymáshoz való viszonyát írja le. Az informatikában az ontológiákat olyan emberek, adatbázisok és alkalmazások használják, akiknek egy bizonyos tudásterületen együtt kell működniük, mint például a zeneipar. Az ontológiák a tudásterület alapfogalmainak, valamint ezek összefüggéseinek géppel értelmezhető definícióit tartalmazzák.

Az ontológiához szorosan kapcsolódik egy fogalom, melyet *taxonómiának* nevezünk. Taxonómia alatt a dolgok hierarchikus formában történő osztályozását értjük. Egy taxonómia ábrázolható fa struktúrában.

Az „ontológia” szót különböző strukturáltsági fokú alkotások megnevezésére használjuk. Ezek az egyszerű taxonómiáktól kezdve (mint pl. a Yahoo hierarchia), a metaadat-sémákon keresztül (mint a Dublin Core), egészen a logikai elméletekig terjedhetnek. A Szemantikus web komoly strukturáltsági fokú ontológiákat igényel, tartalmaznia kell olyan fogalmak specifikációit, mint *osztályok*, *tulajdonságok* és *viszonyok*.

Ha van egy ilyen ontológiánk, akkor ebből logikai következtetéseket lehet levonni, azaz olyan újabb tényeket megállapítani, amelyek explicit módon nincsenek benne az ontológiában, de levezethetők, kikövetkeztethetők belőle a szemantika segítségével. Tehát az ontológiák révén képesek lehetünk automatikus következtetéseket is levonni, ezáltal fejlett szolgáltatásokat nyújtani a modern alkalmazásokhoz.

4.2 Az OWL résznyelvei

A fejlesztők a különböző felhasználói igényeknek megfelelően három alnyelvét dolgozták az OWL nyelvnek, melyek egyre nagyobb kifejezőerőt képviselnek. Ezek sorrendben a következők:

OWL Lite: Ezt az alnyelvet olyan felhasználóknak szánták, akik elsősorban osztályozási hierarchiákat és egyszerű korlátozásokat alkalmaznak. Támogatja a kardinalitáskorlátozást, de a kardinalitás értékeként csak a 0 és 1 értéket engedi meg. Az OWL Lite eszköztámogatása így egyszerűbb lehet, mint a nagyobb kifejező erejű rokonaié.

OWL DL: Ez az alnyelv olyan felhasználók támogatására készült, akik a maximális kifejezőképességet igénylik, mégpedig úgy, hogy a teljes kiszámíthatóság és az eldönthetőség is megmarad, azaz minden konklúzió garantáltan kiszámítható, és minden számítás véges időn belül be is fejeződik. Az OWL DL tartalmazza az OWL összes nyelvi elemét, de ezek csak bizonyos korlátozásokkal használhatók, mint például egy osztály nem lehet egyede egy másik osztálynak. Az OWL DL neve a **D**escription **L**ogics (leíró logika) kifejezésre utal.

OWL Full: Ezt az alnyelvet olyan felhasználóknak szánták, akik a maximális kifejező erőt és az RDF szintaktikai szabadságát igénylik, de ennek fejében lemondanak a kiszámíthatósági garanciákról. Az OWL Full lehetővé teszi egy ontológiának, hogy kiterjessze az előre definiált (RDF vagy OWL) szókészlet jelentéstartományát.

A három alnyelv mindegyike az egyszerűbb elődjének a kiterjesztése abban az értelemben, hogy mit lehet vele legálisan kifejezni, és hogy a velük ábrázolt ontológiából mit lehet érvényesen kikövetkeztetni. Azaz,

- minden legális OWL Lite ontológia egyben legális OWL DL ontológia is,
- minden legális OWL DL ontológia egyben legális OWL Full ontológia is,
- minden érvényes OWL Lite konklúzió egyben érvényes OWL DL konklúzió is,
- minden érvényes OWL DL konklúzió egyben érvényes OWL Full konklúzió is.

Hogyan válasszunk közülük?

Mindig a céljainknak leginkább megfelelő OWL változatot érdemes választani. Az OWL Lite és az OWL DL közötti választás attól függ, hogy a felhasználónak milyen mértékben van szüksége az OWL DL nagyobb kifejező erejű nyelvi konstrukcióira. Az OWL Lite

következtető rendszerei megfelelő számítási tulajdonságokkal fognak rendelkezni, az OWL DL következtető rendszerei, annak ellenére, hogy eldönthető, szélsőséges esetekben túlzottan bonyolultak lehetnek. Az OWL DL és az OWL Full közötti döntés azon múlik, hogy a felhasználó mennyire igényli az új osztályok definiálásának lehetőségét. Amikor OWL Full-t használunk, a következtetések teljes körű szoftvertámogatása az OWL DL-éhez képest kevésbé valószínű.

4.3 OWL alapelemek

Egy OWL ontológia legtöbb eleme az osztályok, tulajdonságok, az osztályok egyedei és a köztük lévő viszonyok leírására használatos. Az OWL névtere a `http://www.w3.org/2002/07/owl#`, prefixe az `owl:`, az ezzel a prefix-szel jelölt elemeket úgy kell értelmezni, hogy azok az OWL névteréből származnak.

4.3.1 Osztályok és egyedek

Az OWL attól függően, hogy milyen módon hoztuk létre, hétféle osztályt különböztet meg.

- Egyszerű nevesített osztályok. (named class)
- Érték-korlátozott osztályok (property restriction)
- Felsorolásos osztályok. (enumeration class)
- Osztályok metszeteként megadott osztályok. (intersection)
- Osztályok uniójaként megadott osztályok. (union)
- Egy osztály komplemenseként megadott osztály. (complement)

A nevesített osztályok az egyetlen, mely rendelkezik URI-val. A többi névtelen, *összetett osztályok*, melyek megkötéseket tesznek arra vonatkozóan, hogy mely egyedek lehetnek a példányaik. Ezeket az OWL *konstruktoraival* képezhetjük. A konstruktorok ún. *osztálykifejezések* előállítására használhatók.

Egyszerű nevesített osztályok

Egy tématerület legáltalánosabb fogalmait olyan osztályoknak kell megfeleltetnünk, amelyek majd a különböző tématerületekhez kapcsolódó taxonómia fák gyökerei lesznek. Az OWL-ban minden egyed az `owl:Thing` (Valami) osztály tagja, valamint minden felhasználó által definiált osztály implicit módon szintén az `owl:Thing` osztály alosztálya. Az OWL definiál

egy üres osztályt is, amelynek a neve `owl:Nothing` (Semmi). Ennek az osztálynak nincs egyetlen példánya sem, és minden osztálynak alosztálya.

```
<owl:Class rdf:ID="MusicArtist"/>
<owl:Class rdf:ID="MusicalManifestation"/>
```

24. példa: Egyszerű nevesített osztályok leírása

Egy osztálydefiníció két részből áll, egy név bevezetéséből vagy ilyenre való hivatkozásból, valamint az osztály korlátozásainak listájából.

Alapvető taxonómiaépítő-elem az osztályok számára a már megismert `rdfs:subClassOf`, melyet a specifikusabb alosztályok leírására használhatunk. Ezt az RDF Sémánál leírt módon alkalmazhatjuk:

```
<owl:Class rdf:ID="Track">
        <rdfs:subClassOf rdf:resource="MusicalManifestation"/>
<owl:Class>
```

25. példa: Egyszerű nevesített osztályok leírása, alosztály korlátozással.

Egyedek

Egy osztály nem más, mint olyan tulajdonságok kollekciója, mely egyedek halmazát írja le. Az egyedek ennek a halmaznak az elemei. Így az osztályoknak meg kell felelniük a dolgok

természetes halmazainak a szóban forgó tématerületen, az egyedeknek pedig meg kell felelniük azoknak a konkrét entitásoknak, amelyek ezekbe az osztályokba becsoportosíthatók.

Egyedeket kétféle módon adhatunk meg, melyek ekvivalensek. Egyrészt az `owl:Thing` elemmel, valamint használhatjuk a már megismert „tipizált csomópont” rövidítést.

Összetett osztályok

A konstruktorok és osztálykifejezések segítségével összetett osztályokat képezhetünk. Az osztálykifejezések egymásba skatulyázhatók anélkül, hogy a közbülső osztályoknak nevet kellene adnunk, ezáltal lehetővé teszi összetett osztályok képzését névtelen osztályokból vagy érték-korlátozott osztályokból.

Felsorolásos osztályok

Az OWL lehetővé teszi, hogy osztályokat specifikáljunk a tagjaik közvetlen felsorolásával, melyet az `owl:oneOf` elem segítségével tehetünk meg. Ez az egyetlen olyan definíció, mely véglegesen specifikálja az osztály egyedeit, azaz továbbiak utólag már nem deklarálhatók az osztály tagjaiként.

Metszet, unió, komplemens

Az OWL támogatja az alapvető halmazműveleteket, nevezetesen az unió, a metszet és a komplementer halmaz képzését, melyekkel manipulálhatjuk az osztályok kiterjedéseit.

Osztályok metszetének megfelelő osztály, az `owl:intersectionOf` konstruktorral képezhetünk, ami azokat az egyedeket választja ki, melyek a felsorolt osztályok közül, valamely osztálynak egyedei.

Az osztályok uniójának megfelelő osztályt, az `owl:unionOf` segítségével konstruálhatjuk. Azok az egyedeket választja ki, melyek a felsorolt osztályok közül, valamely minden egyedei.

Az `owl:complementOf` konstruktor, mely a komplemens halmazképzésnek felel meg, kiválasztja mindazon egyedeket az ábrázolt tématerületről, amelyek nem tartoznak egy bizonyos osztályhoz.

A halmazműveletekkel képzett osztályok tagjaikat teljes mértékben specifikálják.

Diszjunktság

Az osztályok egy adott halmazának a diszjunkt voltát az `owl:disjointWith` konstruktorral fejezhetjük ki. Ez garantálja, hogy egy egyed, amelyik egy osztálynak a tagja, nem lehet egyidejűleg egy másik adott osztálynak is a tagja.

Tulajdonságkorlátozással megadott osztályok

Tulajdonságkorlátozással megadott osztályok olyan névtelen osztályok, amelyekbe tartozó példányok egy megadott tulajdonságkorlátozásnak tesznek eleget. Egy tulajdonságkorlátozás megnevez egy tulajdonságot, majd az értékére vagy a számosságára vonatkozó megkötést tesz. Részletesebben a „további tulajdonságkorlátozások" című részben tárgyalom.

4.3.2 Tulajdonságok

A Tulajdonságok lehetővé teszik számunkra, hogy közös jellemzőket adjunk meg egy osztály összes tagjáról. A tulajdonság bináris reláció, azaz két dolog logikai kapcsolatát leíró fogalom. Az OWL-ban kétfajta tulajdonságtípus van, *adattípus-tulajdonságok* és *objektumtulajdonságok*. Az adattípus-tulajdonságok egyedek és adatok közötti relációk, míg az objektumtulajdonságok az osztályok egyedei között állnak fenn.

Tulajdonságot az `owl:ObjectProperty` elemmel definiálhatunk. Amikor egy tulajdonságot definiálunk meghatározható a tulajdonság érvényességi köre, értéktartománya, vagy a tulajdonság definiálható egy már létező tulajdonság altulajdonságaként. Egy tulajdonsághoz megadható zéró, egy, vagy több érvényességi kör, és értéktartomány korlátozás. Több érvényességi kör megadása azt jelenti, hogy a tulajdonság érvényességi köre a megnevezett osztályok metszete (konjunkciója) lesz, és ugyanez érvényes az értéktartományra is. Ezeket az RDF Sémánál már megismert módon adhatjuk meg. A tulajdonságok, ugyanúgy, mint az osztályok, hierarchiába rendezhetők. Ezt az `rdfs:subPropertyOf` elemmel tehetjük meg. Az altulajdonság örökli a tulajdonságkorlátozásokat.

Adattípus tulajdonságok

Az OWL tartalmaz beépített adattípusokat, melyek az alábbi táblázatban láthatók.

1. táblázat: OWL beépített adattípusok			
xsd:string	xsd:normalizedString	xsd:boolean	
xsd:decimal	xsd:float	xsd:double	
xsd:integer	xsd:nonNegativeInteger	xsd:positiveInteger	
xsd:nonPositiveInteger	xsd:negativeInteger		
xsd:long	xsd:int	xsd:short	xsd:byte
xsd:unsignedLong	xsd:unsignedInt	xsd:unsignedShort	xsd:unsignedByte
xsd:hexBinary	xsd:base64Binary		
xsd:dateTime	xsd:time	xsd:date	xsd:gYearMonth
xsd:gYear	xsd:gMonthDay	xsd:gDay	xsd:gMonth
xsd:anyURI	xsd:token	xsd:language	
xsd:NMTOKEN	xsd:Name	xsd:NCName	
rdfs:Label			

Az adattípus tulajdonságokat az `owl:DatatypeProperty` elemmel definiálhatjuk.

Tulajdonságjellemzők

Az OWL-ban lehetőségünk van a tulajdonságokat tulajdonságjellemzők hozzáadásával tovább specifikálni. Négy féle tulajdonságjellemző létezik:

Tranzitív tulajdonság: Jelölje T(x,y) az x és y között fennálló tulajdonságot. Ha T tulajdonság tranzitív, akkor minden x, y és z-re igaz, hogy

ha T(x,y) és T(y,z) akkor T(x,z).

Megadása: `<rdf:type rdf:resource="&owl;TransitiveProperty" />`

Szimmetrikus tulajdonság: Ha T tulajdonság szimmetrikus, akkor minden x-re és y-ra igaz, hogy

ha T(x,y) akkor T(y,x) és megfordítva.

Megadása: `<rdf:type rdf:resource="&owl;SymmetricProperty" />`

Funkcionális tulajdonság: Ha egy T tulajdonságot funkcionálisnak címkézünk, akkor minden x, y, és z-re igaz, hogy

ha T(x,y) és T(x,z) akkor y = z.

Megadása: `<rdf:type rdf:resource="&owl;FunctionalProperty" />`

Fordított tulajdonság: Ha egy T1 tulajdonságot úgy címkézünk, hogy `owl:inverseOf` T2, akkor minden x-re és y-ra igaz, hogy

ha T1(x,y) akkor T2(y,x) és megfordítva.

Megadása: `<owl:inverseOf rdf:resource="#T2" />`

Fordított funkcionális tulajdonság: Ha egy T tulajdonságot fordított funkcionálisként címkézünk, akkor minden x, y és z-re igaz, hogy

ha T(y,x) és T(z,x) akkor y = z.

Megadása:

```
<rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
```

További tulajdonságkorlátozások

Lehetőség van a tulajdonságok további korlátozására is olyan módon, hogy a tulajdonságkorlátozások nem globálisak lesznek, hanem lokálisak, azaz a tulajdonság nem minden előfordulására vonatkoznak, hanem csak az őket tartalmazó osztálydefinícióra érvényesek. Ezt az `owl:allValuesFrom`, illetve az `owl:someValuesFrom` korátozásokkal tehetjük meg.

Az `owl:allValuesFrom` korlátozás azt köti ki, hogy egy osztály minden egyede számára, a tulajdonság lehetséges értékeit az `owl:allValuesFrom`-al megjelölt osztály összes tagja alkotja.

Az `owl:someValuesFrom` korlátozás azt határozhatjuk meg, hogy egy osztály minden egyedének *legalább egyszer* kell, a tulajdonsággal az `owl:someValuesFrom`-al megjelölt osztályhoz kapcsolódnia.

A két korlátozás közötti logikai különbség nem más, mint az elsőrendű logika univerzális és egzisztenciális kvantora közötti különbség.

Az `owl:Cardinality` lehetővé teszi, hogy korlátozzuk egy tulajdonság kardinalitását, azaz hogy pontosan megadjuk egy adott reláció előfordulásainak a számát egy adott osztály esetén.

Az `owl:hasValue` tulajdonságkorlátozás lehetővé teszi, hogy osztályokat definiáljunk egy meghatározott tulajdonságérték megléte alapján. Ebből következően egy egyed ennek az osztálynak a tagja lesz, ha a tulajdonság értékeinek legalább egyike egyenlő az `owl:hasValue` kifejezésben megadott értékkel.

Ha egy osztály példányai eleget tesznek ezeknek a tulajdonságkorlátozásoknak, akkor az osztályt tulajdonságkorlátozással létrehozott osztálynak nevezzük.

4.4 Ontológiák szerkezete, egyesítése

A szemantikus web a jellegéből adódóan elosztott rendszer, az OWL-nak lehetővé kell tennie, hogy az információkat gyűjthessünk elosztott forrásokból is. Az OWL nyitott világot feltételez, vagyis az erőforrások leírásai nem korlátozódnak egyetlen fájlra vagy területre. Emiatt az OWL lehetővé teszi az ontológiák összekapcsolását, mely felvet néhány problémát. Az ontológiák egyesítésénél az új információk ellentmondásosak lehetnek, azonban tények és következmények mindig csak hozzáadhatók, sohasem törölhetők. Ezért szükség van olyan eszközök beépítésére is melyekkel kiküszöbölhetőek ezek az ellentmondások.

Egyenértékűség osztályok között

Ahhoz, hogy egy sor meglévő ontológiát komponensként egy újabb ontológiába beépíthessünk, gyakran meg kell jelölnünk, hogy az egyik ontológiában egy adott osztály vagy tulajdonság egyenértékű egy másik ontológia valamelyik osztályával vagy tulajdonságával.

Az `owl:equivalentClass` tulajdonságot használjuk annak jelölésére, hogy a két osztálynak pontosan ugyanazok az egyedei.

Egyenértékűség tulajdonságok között

A tulajdonságok az osztályokhoz hasonló módon történő összekapcsolására az `owl:equivalentProperty` konstruktort használjuk.

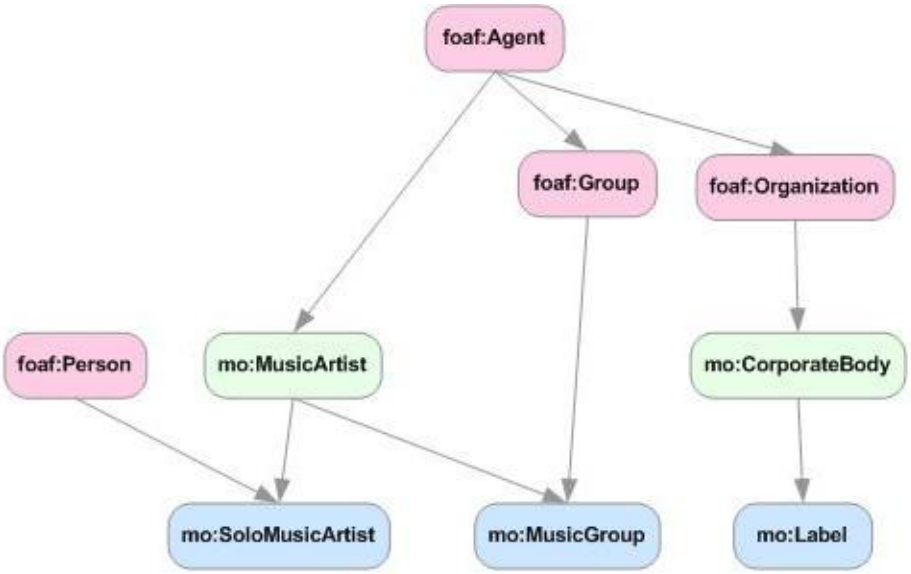
Egyedek azonossága

Az OWL nem feltételez egyedi neveket, így tehát ha csak a nevek különböznek, még nem jelenti azt, hogy ezek különböző egyedekre is hivatkoznak. Az `owl:sameAs` konstruktorral jelenthetjük ki két egyedről, hogy azonosak.

Egyedek különbözősége

Vannak olyan esetek, amikor garantálnunk kell különböző entitások kölcsönös különbözőségét. Az `owl:sameAs` hatásának a fordítottját az `owl:differentFrom` használatával érhetjük el. Egy alkalmasabb mechanizmus az `owl:AllDifferent`, amellyel kölcsönösen különböző egyedek halmazát lehet definiálni.

Következzen egy részlet a „Music Ontology” (`http://musicontology.com`) ontológiából az elmondottak illusztrálására:



8. ábra: A mo:MusicArtist, mo:MusicGroup, és mo:Organization sémák

(forrás: http://wiki.musicontology.com/index.php/Classes_Schemas)

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:vs="http://www.w3.org/2003/06/sw-vocab-status/ns#"
        xmlns:time="http://www.w3.org/2006/time#"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:mo="http://purl.org/ontology/mo/">
<!— ONTOLÓGIA DEFINÍCIÓ (ONTOLOGY DEFINITION) -->
<owl:Ontology rdf:about="">
        <dc:title>The Music Ontology</dc:title>
        <owl:versionInfo>Revision: 1.03 - 2</owl:versionInfo>
        <dc:description>
           A Music Ontology specifikáció fő fogalmakat és
           tulajdonságokat szolgáltaltat zenék (pl. előadók, albumok)
           leírására a Szemantikus weben. Ez a dokumentum a Music
           Ontology részletes leírását tartalmazza.
          (The Music Ontology Specification provides main concepts and
           properties for describing music (i.e. artists, albums)
           on the Semantic Web. This document contains a detailed
           description of the Music Ontology.)
        </dc:description>
        <dc:date>$Date: 2007/03/20 11:56:00 $</dc:date>
</owl:Ontology>
```

```
<!– A kifejezések státusza (Status of terms) -->

<owl:AnnotationProperty rdf:about="http://www.w3.org/2003/06/sw-vocab-
status/ns#term_status"/>

<!– OSZTÁLYOK (CLASSES) -->

<!–A foaf:Agent –hez kapcsolódó osztályok
        (Classes related to foaf:Agent) -->

<rdfs:Class rdf:about="http://purl.org/ontology/mo/MusicArtist"
                rdfs:label="MusicArtist" vs:term_status="stable">
        <rdfs:comment>
                Személy vagy emberek csoportja (vagy számítógép),
                aki(k)nek a kreatív zenei munkája érzékenységet és
                képzelőerőt mutat.
                (A person or a group of people (or a computer :-) ),
                 whose musical creative work shows sensitivity and
                 imagination)
        </rdfs:comment>
        <rdfs:subClassOf
                 rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
        <rdfs:isDefinedBy rdf:resource="http://purl.org/ontology/mo/"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdfs:Class>

<rdfs:Class rdf:about="http://purl.org/ontology/mo/SoloMusicArtist"
        rdfs:label="SoloMusicArtist" vs:term_status="stable">
        <rdfs:comment>
                Egyedülálló személy, akinek a kreatív zenei munkája
                érzékenységet és képzelőerőt mutat.
                (Single person whose musical creative work shows
                sensitivity and imagination.)
        </rdfs:comment>
        <rdfs:subClassOf rdf:resource=
                "http://purl.org/ontology/mo/MusicArtist"/>
        <rdfs:subClassOf rdf:resource=
                "http://xmlns.com/foaf/0.1/Person"/>
        <rdfs:isDefinedBy rdf:resource=
                "http://purl.org/ontology/mo/"/>
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdfs:Class>
...
```

```
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about=
                "http://purl.org/ontology/mo/engineered"/>
        </owl:onProperty>
        <owl:someValuesFrom rdf:resource=
                "http://purl.org/ontology/mo/Performance"/>
      </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Agent"/>
 </owl:Class>
...
```

(forrás: http://pingthesemanticweb.com/ontology/mo/musicontology.rdfs)

5 A szemantikus web elképzelés alkalmazása XML alapú rendszerekben

5.1 RDF adatok elhelyezése

Az RDF egy általános keretrendszer, amely segítségével metainformációkat társíthatunk erőforrásokhoz. Az RDF bizonyos értelemben nem is kötődik a Webhez, az erőforrás-azonosítás az URI-k feladata. RDF-et használhatunk például a Webtől független adatbázisok esetén, intelligens keresésére, következtetésre is. Általánossága ellenére az RDF felhasználásának egyik elvárt módja, hogy megteremtse a szemantikus web alapjait. Ehhez nyilvánvalóan szükséges, hogy az RDF-ben tárolt információk hozzáférhetőek legyenek a Weben, és a keresőrendszerek el tudják érni azt. Ebből természetes módon következik, hogy érdemes lenne az RDF alakú metaadatokat egy honlap részeként, vagy ahhoz kapcsolva megadni, mivel a keresőrendszerek robotjai már amúgy is periodikusan járják a Webet. A szokásos gyűjtögetésük során hozzájuthatnának ahhoz a többlet háttértudáshoz, melyek a felhasználók kérdéseinek pontosabb megválaszolásához szükségesek.

5.1.1 RDF HTML-be ágyazva

Az előző részekben bemutattam, hogy az RDF-állítások szerializálhatóak szabványos XML alakban, melynek eszköze az RDF/XML. Mivel az RDF/XML leírások szintaktikailag közönséges, jól formázott XML-állományok, az egyszerű XML-elemzők változtatás nélkül képesek azok beolvasására. A HTML XML-alapú változata az XHTML, mely egyre nagyobb teret nyer a Weben. Lévén, hogy az XHTML és az RDF/XML is XML-alapú dokumentum, és emiatt nagymértékben integrálhatók, az RDF XHTML-be ágyazása úgymond „adja magát”.

Az RDF adatok elhelyezésére egy HTML-dokumentumban a legideálisabb rész az oldal feje, azaz a `<HEAD>` és `</HEAD>` közötti rész, mivel szinte biztosra vehető, hogy a keresőrobotok beolvassák ezt a részt, hiszen a HTML által biztosított alapszintű információk is itt találhatók, mint például a honlap címe. Az RDF-állítások leírásához az egyik egyszerűsített írásmódot kell használnunk, amely lehetővé teszi, hogy az olyan RDF predikátum párokat, ahol a tárgy literál, XML-attribútumértékként adjunk meg. Erre azért van szükség, mert a legtöbb böngésző megjeleníti az egyszerű XML-elemek tartalmát, függetlenül attól, hogy ismeri-e az adott elemet vagy sem.

```
<html>

   <head>
   <title>Weboldal metaadatokkal<title>
   <meta http-equiv=”content-type”
         content=”text/html; charset=utf-8”/>

   <rdf:RDF>
        ...RDF leírások...
   </rdf:RDF>

   </head>

   <body>
        ...Tartalom...
   </body>

</html>
```

Sajnos az RDF ily módon történő weboldalakba ágyazása korántsem tökéletes, sok probléma merülhet fel. Ezek közül talán a legfontosabb, hogy beágyazott RDF adatot tartalmazó XHTML-állomány **nem érvényes** dokumentum, mert nem tesz eleget az XHTML sémának. Ez érthető is mivel olyan RDF specifikus elemeket tartalmaz, melyekre az XHTML specifikáció nem, és nem is hivatott felkészülni. Ennek ellenére sok jelenlegi RDF-elemző és feldolgozó alkalmazás támogatja az RDF-metatadatok XHTML-ből történő kinyerését.

5.1.2 RDF HTML-hez kapcsolása

Egy másik gyökeresen különböző elképzelés, hogy weboldalakhoz ne úgy társítsunk RDF-metaadatot, hogy azt a honlap részévé tesszük, hanem csak egyszerűen adjunk egy a metaadatokra mutató linket. Ezt a linket a keresőrobotok ugyanúgy fogják követni, mint bármilyen más linket az oldalunkon, és így eljutnak magához az RDF-leírásunkhoz is.

Ha így helyezzük el RDF-forrásainkat, nem lépnek fel a beágyazásnál tapasztalt problémák, könnyebb, weblaptól független a karbantartás, módosítás. Architektúrális szempontból ez a legtisztább megoldás, azonban nem használja ki az RDF/XML azon előnyét, hogy XML-alapú. A módszer előnye azonban, hogy más szerializációs eljárással készített RDF leírásokat is hozzákapcsolhatunk weblapunkhoz, nem szükséges a néhol bonyolult RDF/XML-t használnunk.

A megfelelő linket több helyen is elhelyezhetjük. Az egyik módszer az oldalunk fejében a `<link>` elem használatával, melynek feladata hogy összekapcsolja oldalunkat külső erőforrásokkal.

Például:

```
...
<head>
   <title>Weboldal metaadatokkal</title>
   <link rel=”meta” type=”application/rdf+xml”
        href=”metadatok.rdf”/>
</head>
...
```

28. példa: RDF állomány csatolása a `<link>` elemmel

A társításhoz használhatjuk a közönséges `<a>` elemet is, az oldalunk tetszőleges részén.

Például:

```
...
<body>
   <h1>
      <a rel=”meta” type=”application/rdf+xml”
        href=”metadatok.rdf”>
      Hello Világ!
      </a>
   </h1>
</body>
...
```

29. példa: RDF állomány csatolása az `<a>` elemmel

5.1.3 RDFa

Az RDF-adatok XHTML-be ágyazásának másik módja, az RDFa szerializációs szintaxis használata, melyet W3C dolgozott ki, és 2008 óta tartozik a hivatalos ajánlásai közé. Használata a HTML4-es verziója óta támogatott.

Az RDFa előnye, hogy XTHML attribútumokat használ az RDF-kijelentések leírására, emiatt használata könnyebb az RDF/XML szintaxisnál, és nem eredményezi annak problémáit sem. Meg kell jegyezni azonban, hogy jelenleg az RDFa nem validált a HTML4-ben, az ezt használó XHTML-dokumentumok „csak” az XHTML1-et kiterjesztő XHTML1.1+RDFa DTD sémának megfelelően nevezhetők érvényesnek.

RDFa attribútumok

Az RDFa-ban használatos attribútumok közül, néhány régebbi XHTML-attribútum, és van néhány új, RDFa specifikus attribútum is. A következőkben ezeket veszem sorra, bemutatva mi lehet az értékük és mire használatosak.

A régebbiek:

@rel: whitespace karakterekkel határolt relatív URI-k listája, két erőforrás közötti reláció kifejezésére.

@rev: whitespace karakterekkel határolt relatív URI-k listája, két erőforrás közötti fordított reláció kifejezésére.

@content: egy sztring, a literál tartalmának gépi olvasásra alkalmassá tételére (plain literal object)

@href: URI a partner erőforrás kapcsolat kifejezésére (resource object)

@src: URI a partner erőforrás kapcsolat kifejezésére, ha az erőforrás beágyazott (resource object)

RDFa specifikusak:

@about: URI a kijelentés alanyának azonosítására (subject)

@property: whitespace karakterekkel határolt relatív URI-k listája, a kijelentés állítmányának azonosítására (predicate)

@resource: URI a kijelentés tárgyának azonosítására (object)

@datatype: relatív URI, a típusos literálok típusának megadásához

@typeof: whitespace karakterekkel határolt relatív URI-k listája, segítségével az RDF kijelentés alanyához RDF-típusokat rendelhetünk.

Lássunk néhány példát a használatukra:

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  >
  <head>
    <title>Kispál és a Borz</title>
    <meta property="dc:creator" content="Fejlesztő Aladár" />
    <link rel="foaf:topic" href="http://www.weblapbolt.hu/#fa" />
  </head>
  <body>...</body>
</html>
```

30. példa: RDFa weboldal fejrészében

Az RDFa kijelentéseket elhelyezhetjük a weblap fejrészében. A fenti példában az oldal készítőjét és annak FOAF profiját is megadtuk.

Közkedvelt módszer az RDFa-ban, hogy a leírásokat a `<span>` elem attribútumaiként adjuk meg, mivel ez az elem a dokumentum törzsében bárhol elhelyezhető.

```
...
<span about="http://www.kispalesaborz.hu "
   property="dc:title" content=" Kispál És a Borz "/>
...
<span about="http://www.kispalesaborz.hu"
   property="dc:title">Kispál És a Borz</span>
...
```

31. példa: RDFa a weboldal törzsében

A kétféle leírás szintaktikailag különböző, de szemantikailag ugyanazt az állítást fogalmazza meg, mely szerint a "`http://www.kispalesaborz.hu`" weboldal neve „Kispál és a Borz”.

Lehetőségünk van több kijelentést ugyanarról az erőforrásról rövidített formában is megadni.

```
<img src="http://www.kispalesaborz.hu/image.php?id=2318"
     about=" http://www.kispalesaborz.hu/image.php?id=2318">
  <span property="dc:format" content="img/jpeg"/>
  <span property="dc:creator" content="Fényképész Kázmér"/>
  <span property="dc:subject" content="Kispál és a Borz a Lovardában"/>
</img>
```

32. példa: Több kijelentés ugyanarról az erőforrásról

A kijelentések beágyazhatóak a weblap szövegébe is.

```
<p>Ezt a fotót <span class="author" about="koncert001.jpg"
property="dc:creator">Fényképész Kázmér</span> készítette.</p>
```

33. példa: RDFa szövegbe ágyazva

5.2 Néhány RDF alkalmazás a gyakorlatban

Dolgozatom előző részeiben bemutattam a szemantikus web alapjaiul szolgáló RDF-et, az OWL-t. Megmutattam hogyan lehet ezeket weboldalak kontextusában elhelyezni. A következőkben szeretnék megismertetni néhány igazi, gyakorlati RDF alkalmazást, hogy bemutassam, miként támogatja az RDF a való világ követelményeit a legkülönfélébb dolgokról szóló információk ábrázolásában és manipulálásában.

5.2.1 Dublin Core

A Dublin Core tulajdonképpen "elemek" (tulajdonságok) egy halmaza, dokumentumok metaadatokkal történő leírása. Az elemek halmazát eredetileg az Ohio állambeli Dublin-ban,

1995-ben megrendezett első Meta-adat Műhelykonferencián fejlesztették ki, majd későbbi ilyen rendezvények során tovább tökéletesítették, jelenleg pedig a Dublin Core Metadata Initiative nevű szervezet tartja karban. A Dublin Core célja szabványos adatleíró elemek egy olyan minimális halmazának a biztosítása, amely lehetővé teszi az Interneten tárolt, dokumentum jellegű objektumok leírását és automatikus indexelését, a könyvtári kártyakatalógusok analógiájára.

A Dublin Core jelenlegi elemeit a DC Metaadat Elemhalmaz definiálja, mely az alábbi tulajdonságdefiníciókat tartalmazza:

- **Title** (Címe): A dokumentum jellegű erőforrás neve.
- **Creator** (Szerzője): Az az entitás (személy vagy szervezet), mely elsődlegesen felelős az erőforrás tartalmának elkészítéséért.
- **Subject** (Témája): Az erőforrás tartalmának a tárgya.
- **Description** (Leírása:) Az erőforrás tartalmának összefoglalása.
- **Publisher** (Kiadója): Az az entitás (személy vagy szervezet), mely felelős az erőforrás nyilvánosságra hozásáért.
- **Contributor** (Társszerzője): Egy olyan entitás (személy vagy szervezet), amely munkájával hozzájárult az erőforrás tartalmának létrehozásához.
- **Date** (Dátuma): Az erőforrás életciklusának valamely eseményéhez kapcsolódó időpont.
- **Type** (Típusa): Az erőforrás tartalmának természete, jellege.
- **Format** (Formátuma): Az erőforrás fizikai vagy digitális megjelenési formája.
- **Identifier** (Azonosítója): Egy egyértelmű hivatkozás az erőforrásra az adott kontextuson belül.
- **Source** (Forrás): Hivatkozás egy olyan erőforrásra, ahonnan a jelenlegi erőforrás eredetileg származik.
- **Language** (Nyelve): Az a nyelv, amelyen az erőforrás szellemi tartalma megjelenik.
- **Relation** (Kapcsolata): Hivatkozás egy, a leírt erőforráshoz kapcsolódó másik erőforrásra.
- **Coverage** (Lefedése): Az erőforrás tartalmának tematikai/területi átfogása.
- **Rights** (Szerzői és egyéb jogok): Információ azokról a jogokról, amelyek az erőforráshoz fűződnek, vagy rajta keresztül keletkeznek

A Dublin Core (DC) elemeket használó információkat bármilyen alkalmas nyelven megadhatjuk, azonban az RDF ideális eszköz a DC információ ábrázolására. A DC szókészlet névtér URI-je a `http://purl.org/dc/elements/1.1/`, prefixként a `dc:` -t szokás alkalmazni. A dolgozatom előző példáiban magam is felhasználtam a DC elemeket, például a 13. példában.

5.2.2 XMP

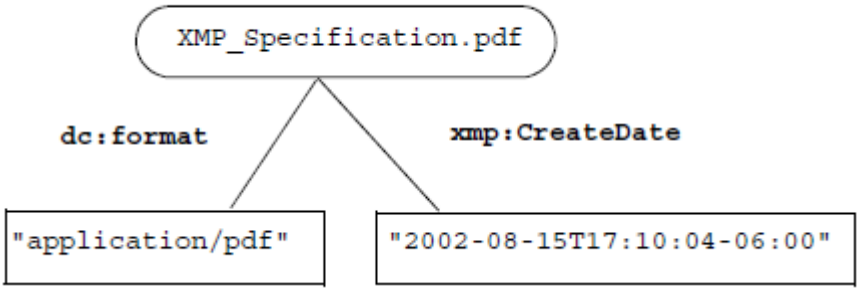
Az Adobe cég RDF alapú XMP platformja (**E**xtensible **M**etadata **P**latform) jó példa egy olyan technológiára, mely lehetővé teszi egy fájl leírására használt metainformációk beépítését magába a fájlba. Az XMP a metaadatok ábrázolásának alapjául az RDF/XML-t használja. A metaadatokat XMP csomagokba lehet beágyazni, amelyek „becsomagolják" az RDF/XML-ben leírt metaadatokat. Számos elterjedten használt állományformátumhoz (GIF, JPEG, PDF, PNG, PostScript, TIFF) leírt, hogy a beágyazás hogyan végezhető el.

Előnyei, hogy

- szabványos és állományformátumtól független módját adja digitális képek és egyéb erőforrások metaadatokkal történő annotálásának,
- átvitel során a metaadatok a beágyazó erőforrással együtt utaznak, soha nem vesznek el útközben,
- az egységesen ábrázolt metaadatok olyan alkalmazások számára is elérhetők, amelyek nem feltétlenül ismerik a beágyazó erőforrás formátumát,
- új dimenziókat nyit a digitális fotózásban, és a képszerkesztő alkalmazások számára,
- ha széles körben támogatott lesz, valamint elterjedten használt a Weben, metaadatok hatékonyan kiaknázható forrását biztosítja a szemantikus web alkalmazások számára

Ma már számos Adobe termék támogatja az XMP-t.

Tekintsük az alábbi egyszerű példát, mely egy PDF dokumentumról állítja, hogy formátuma pdf, valamint leírja létrehozásának idejét:



9. ábra: Két egyszerű XMP állítás

(forrás: http://www.adobe.com/devnet/xmp/pdfs/XMPSpecificationPart1.pdf)

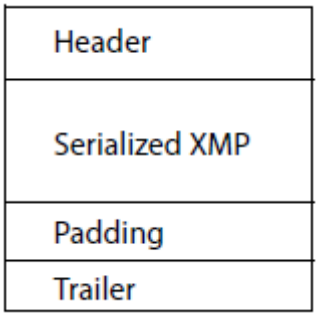
Ezt a két állítást a következőképpen szerializálhatjuk RDF/XML-ben:

```
<rdf:RDF xmlns:rdf= ...>
   <rdf:Description rdf:about=""
            xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:format>application/pdf</dc:format>
   </rdf:Description>
   <rdf:Description rdf:about=""
            xmlns:xmp="http://ns.adobe.com/xap/1.0/">
      <xmp:CreateDate>2002-08-15T17:10:04Z</xmp:CreateDate>
   </rdf:Description>
</rdf:RDF>
```

34. példa: A fenti gráf RDF/XML szerializációja

(forrás: http://www.adobe.com/devnet/xmp/pdfs/XMPSpecificationPart1.pdf)

Ezt a leírást ún. XMP csomagok formájában csatolhatjuk magához a fájlhoz. Ennek sémája a következő:



10. ábra: XMP csomag sémája

(forrás: http://www.adobe.com/devnet/xmp/pdfs/XMPSpecificationPart3.pdf)

XML formája:

```
<?xpacket begin="■" id="W5M0MpCehiHzreSzNTczkc9d"?>
... a fentebb leírt szerializált XMP : ...
   <x:xmpmeta xmlns:x="adobe:ns:meta/">
      <rdf:RDF xmlns:rdf= ...>
            <rdf:Description rdf:about="..."
                    xmlns:dc="http://purl.org/dc/elements/1.1/">
               <dc:format>application/pdf</dc:format>
            </rdf:Description>
            <rdf:Description rdf:about="..."
                    xmlns:xmp="http://ns.adobe.com/xap/1.0/">
            <xmp:CreateDate>2002-08-15T17:10:04Z</xmp:CreateDate>
   </rdf:Description>
      </rdf:RDF>
   </x:xmpmeta>
... XML whitespace as padding ...
<?xpacket end="w"?>
```

5.2.3 PRISM

A folyóiratok kiadásával foglalkozó szervezetek is kidolgozták a maguk metaadat-specifikációját, melynek neve PRISM (**P**ublishing **R**equirements for **I**ndustry **S**tandard **M**etadata). A kiadók sokféle módon szeretnék felhasználni a meglévő tartalmat annak érdekében, hogy jobban megtérüljenek a befektetéseik. Ilyenek például a cikkek publikációs licenceinek eladása, a publikált tartalom újrafelhasználása, például könyv formában történő kiadása, stb. Hogy ezeket a célokat megvalósítsák, egy olyan metaadat rendszerre van szükségük, amely a hangsúlyt az erőforrások felfedezésére, a jogok nyomon követésére, valamint a meta-adatok folyamatos megőrzésére fekteti.

Az *erőforrások felfedezése* egy általános kifejezés a tartalom megtalálásának folyamatára, mely magában foglalja a keresést, a böngészést, a tartalom irányítását és más technikákat is. A PRISM, hogy növelje a felfedezés valószínűségét, olyan tulajdonságokat definiál, amelyekkel leírható az erőforrások témája, formátuma, jellege, eredete és kontextusa. De eszközöket biztosít az erőforrások kategorizálására is, tematikus taxonómiák formájában.

A *jogok nyomon követése* nagyon fontos, hiszen a magazinok gyakran vesznek át anyagokat más magazinoktól, kiadói licencek alapján. Ezért a PRISM rendszer olyan elemeket tartalmaz, amelyekkel nyomon követhetők az efféle jogok. A RRISM specifikációja egy külön szókészletet definiál azoknak a helyeknek, időpontoknak és kiadói szakmáknak a leírására, ahol az adott tartalom felhasználható, illetve, ahol nem felhasználható.

Ahogy a tartalom mozog a rendszerek között, a metaadatokat gyakran eldobják, hogy azután a produkciós folyamat során, jelentős költséggel újra előállítsák azokat. A PRISM megpróbálja csökkenteni ezt a problémát azzal, hogy olyan metaadat-specifikációt készít, amelyik használható a tartalom előállítási folyamatának minden fázisában, ezzel elősegítve a *metaadatok megőrzését*.

A PRISM specifikációjának fontos jellemzője a meglévő specifikációk alkalmazása. A PRISM felhasználja az XML, az RDF, a Dublin Core specifikációit, valamint a különböző ISO szabványú formátumokat és szókészleteket. Tovább bővíti a Dublin Core-t is abból a célból, hogy még részletesebb leírásokat lehessen készíteni vele. A bővítményeket három új szókészletben definiálja, ezek a következők:

A `prism:` minősítettnév-prefix a PRISM fő szókészletét azonosítja, amelyben a kifejezések a `http://prismstandard.org/namespaces/basic/1.0/` URI-prefixet használják. Ebben a szókészletben a legtöbb tulajdonság a Dublin Core tulajdonságok specifikusabb változata.

A `pcv:` (PRISM Controlled Vocabulary) prefix a PRISM "szabályozott" (controlled) szókészletét azonosítja. Ennek a szókészletnek a kifejezései mind a `http://prismstandard.org/namespaces/pcv/1.0/` URI prefixet használják. A szókészlet olyan tulajdonságokat tartalmaz, amelyekkel kifejezéseket definiálhatunk, leírhatjuk a köztük lévő viszonyokat, és alternatív változatokat is kapcsolhatunk hozzájuk.

A `prl:` (PRISM Rights Language) nevű szókészlet kifejezései mind a `http://prismstandard.org/namespaces/prl/1.0/` URI prefixet használják. Ez a digitális jogok kezelésére szolgáló szókészlet.

A PRISM rendszer azért használja az RDF-et, mert ez lehetővé teszi a legkülönbözőbb komplexitású leírások kezelését is, az egyszerűbb karakterláncokat és kulcsszavakat használóktól, a sokkal strukturáltabb metaadat leírásokig.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:pcv="http://prismstandard.org/namespaces/pcv/1.0/"
         xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description rdf:about="
        http://www.kispalesaborz.hu/image.php?id=2318">">
    <dc:creator>
      <pcv:Descriptor rdf:about="http://koncertfoto.hu/emp3845">
        <pcv:label>Fényképész Kázmér</pcv:label>
      </pcv:Descriptor>
    </dc:creator>
    <dc:coverage>
      <pcv:Descriptor
          rdf:about="http://prismstandard.org/vocabs/ISO-3166/HU">
        <pcv:label xml:lang="hu">Magyarország</pcv:label>
        <pcv:label xml:lang="en">Hungary</pcv:label>
      </pcv:Descriptor>
    </dc:coverage>
  </rdf:Description>
</rdf:RDF>
```

területe (coverage) Magyarország.

5.2.4 XPackage

Az XML Csomag specifikáció (XML Package vagy XPackage) egy keretet biztosít a csomagoknak nevezett csoportosítások definiálásához, és a bennük szereplő erőforrások leírásához, azaz az erőforrások tulajdonságainak, a beépítésük módjának és egymáshoz való viszonyának a leírásához. Az XPackage alkalmazásai specifikálják egy dokumentum által használt stíluslapokat (style sheets), a dokumentum által használt képeket és ábrákat, a dokumentumok szerzőjét és más metaadatait. Tartalmazza annak leírását, hogy miként használják a névtereket az XML erőforrások, valamint egy „csomagjegyzéket", amely leírja, hogy milyen erőforrásokat csomagoltunk össze egyetlen archív-fájlba. Az XPackage keretrendszer az XML-re, az RDF-re és az XML Linking Language-re épül.

Az XPackage egyik alkalmazása az XHTML dokumentumok és támogató erőforrásaik leírása, majd ezek egy RDF alapú csomagleíró dokumentumban való tárolása. Az erőforrások leírása a csomagleíró dokumentumokban szabványos RDF/XML szintaxissal történik. Több RDF szókészletet tartalmaz, egyet az általános összecsomagolási leírások számára, és több szókészletet az erőforrásokról szóló kiegészítő információk megadásához, ezek a következők:

- A `file:` prefixet használó szókészlet a fájlok leírására alkalmas.
- A `mime:` prefixet használó szókészlet MIME információk leírására használhatjuk.
- Az `unicode:` prefixet használó szókészletet használhatjuk a karakterhasználati információk megadására.
- Az `x:` prefixet használó szókészlet XML alapú erőforrások leírására használatos.

5.2.5 RSS 1.0

Ahogy az erőforrások mennyisége és változatossága növekszik a Weben, egyre nehezebb lesz kezelni és egésszé integrálni az ilyen információkat. Az RDF Webhely-összefoglaló (**R**DF **S**ite **S**ummary: RSS 1.0) egy olyan RDF szókészlet, mely egy könnyű, de hatékony lehetőséget ad az olyan információk leírására és újrafelhasználására, amelyeket meghatározott időben, nagy tömegben kell az érdekeltekhez eljuttatni. A formátumot úgy tervezték meg, hogy a tartalmat könnyen megkülönböztethető szekciókba lehessen csomagolni. A hírportálok, a blogok, a sporteredmények, a tőzsdei hírek, és ehhez hasonló információküldő szolgáltatások az RSS 1.0. tipikus alkalmazási esetei. Az utóbbi időben az RSS 1.0 alkalmazások kezdenek három különböző kategóriára bomlani:

On-line tartalomegyesítő webhelyek: Ezek a weboldalak források ezreiből gyűjtik az automatikusan küldött anyagokat, és először szétszedik `<item>` tegek szerint, majd pedig nagy, tematikus csoportokba ismét összerakják őket, végül az ilyen csoportot kereshetővé teszik. Ilyen módon megkereshetjük a több száz webhelyről származó legfrissebb híreket, anélkül hogy egyet is fel kellene keresnünk.

Asztali hírolvasó programok: Lehetővé teszik a felhasználóiknak, hogy akár az otthoni számítógépükről előfizessenek akár több száz automatikusan küldött információra is, és így naprakészek maradjanak egy-egy adott témában.

Inzertek (scripts): Az RSS eredeti célja az volt, hogy lehetővé tegye a más webhelyekről származó információk beépítését saját weboldalunkba.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:content="http://purl.org/rss/1.0/modules/content/"
        xmlns:tourtracker="http://www.tourtracker.com/"
        xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/" version="2.0">
  <channel>
    <title>
        Tour Tracker Events
    </title>
    <link>
        http://www.tourtracker.com
    </link>
    <description>
        Tour Tracker Events for Metallica
    </description>
    <item>
      <title>
                Metallica playing at Telenor Arena
      </title>

      <link>
                http://www.tourtracker.com/shows/metallica/telenor-arena/
                04-13-2010/1109842-1004203
      </link>
      <description>
                &lt;p&gt;Metallica is performing at &lt;p&gt;
                Telenor Arena in Oslo, NOR on Tue, 13 Apr 2010
      </description>
      <guid>
                http://www.tourtracker.com/shows/metallica/telenor-arena/
                04-13-2010/1109842-1004203
      </guid>
      <tourtracker:title>
                Metallica playing at Telenor Arena
      </tourtracker:title>
      <tourtracker:venue>
                Telenor Arena
      </tourtracker:venue>

      <tourtracker:eventDate>
                04-13-2010
      </tourtracker:eventDate>
    </item>
    <item>
        ...
    </item>
...
</channel>
</rss>
```

37. példa: RSS 1.0, a Metallica turnéi

5.2.6 FOAF

A FOAF (**F**riend **O**f **A** **F**riend) egy nyílt, decentralizált technológia személyek és csoportok leírására. A project célja az emberek leírhatók legyenek a „gépek számára is olvasható” formában. Leírhatjuk az emberek között fennálló kapcsolatokat, ezzel létrehozhatunk embercsoportok között fennálló szociális hálókat anélkül, hogy egy központi tárolóhelyre szükségünk lenne. Leírhatjuk az emberek kapcsolatait a világ objektumaival, olyan dolgokat, amiket létrehoznak vagy tesznek. Emberek ezrei nap, mint nap adnak leírást magukról weboldalukon, a FOAF segítségével úgy tehetjük ezt meg, hogy az információk a gépek és keresők számára is érthető legyen. Például, egy a FOAF profilokat használó program képes lehet a kapcsolatokon keresztül, kilistázni Európa összes lakosát. A technológia nagyon egyszerű, nagymértékben támogatja az információk megosztását webhelyek között. A profilokban tárolt metaadatok automatikusan kiterjeszthetőek vagy összefésülhetőek. A FOAF egy ontológia, mely az RDF és az OWL technológiákat használja fel, saját leíró szókészletének definiálásához. A szókészlet elemei `foaf:` prefixet használják, melynek URI-ja a `http://xmlns.com/foaf/0.1/`.

```
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:admin="http://webns.net/mvcb/">
<foaf:PersonalProfileDocument rdf:about="">
  <foaf:maker rdf:resource="#me"/>
  <foaf:primaryTopic rdf:resource="#me"/>
  <admin:generatorAgent rdf:resource="http://.../foaf-a-matic"/>
  <admin:errorReportsTo rdf:resource="mailto:leigh@ldodds.com"/>
</foaf:PersonalProfileDocument>
<foaf:Person rdf:ID="me">
<foaf:name>Péter Példa</foaf:name>
<foaf:title>Mr</foaf:title>
<foaf:givenname>Péter</foaf:givenname>
<foaf:family_name>Példa</foaf:family_name>
<foaf:nick>Peti</foaf:nick>
<foaf:mbox_sha1sum>...</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="http://myhomepage.com"/>
<foaf:depiction rdf:resource="http://myhomepage.com/me.jpg"/>
<foaf:schoolHomepage rdf:resource="http://www.unideb.hu"/>
<foaf:knows>
<foaf:Person>
<foaf:name>Panna Példa</foaf:name>
<foaf:mbox_sha1sum>a1e4e5e1971739c90dc8a27b3457b4c9e002db6f</foaf:mbox_
sha1sum></foaf:Person></foaf:knows></foaf:Person>
</rdf:RDF>
```

a komplexitásából származó hátrányait igyekeztek kiküszöbölni.

Egy-egy metaadattal nem rendelkező zenei állomány lejátszásakor felcsatlakozhatunk az MB szerverre és lekérdezhetjük a szükséges információkat. A MusicBrainz adatokat tárol a zenészekről, a műveikről, és az ezek közti kapcsolatokról. A zeneművekről minden esetben összegyűjti az album címét, a számok címeit, és az egyes számok hosszúságát. Ezeket az adatokat központi, formai és egyéb irányelvek szerint tartják karban. Opcionálisan tárolható az egyes albumok kiadási dátuma és helye, a CD azonosítója, az egyes számok akusztikus ujjlenyomata, és tetszőleges megjegyzések is. Jelenleg a MusicBrainz adatbázisa kb. 534 000 előadót, 800 000 albumot és mintegy 9,3 millió zeneszámot tartalmaz.

A sémában használt fő osztályok az előadó (*artist*), kiadvány (*release*), zeneszám (*track*) és címke (*label*). Mindegyikhez különböző attribútumok (*attributes*) és kapcsolatok (*relations*) csoportja tartozik. A hagyományos kapcsolatokon kívül (mint például az előadók kiadványokat adnak ki, a kiadványokon zeneszámok vannak), leírhatunk speciálisabb kapcsolatokat is, az erre szolgáló sémát „*AdvancedRelationships*”-nek nevezik.

Az Artist (előadó) osztály: Az előadók mindig rendelkeznek egyedi azonosítóval (*unique ID*), névvel (*name*) és egy rövid névvel (*sort name*). Ha az előadó neve nem egyedi, akkor használhatunk egy „egyértelműsítő” megjegyzést (*disambiguation* comment), mellyel további információkat adhatunk róla. Továbbá, az előadókat megjelölhetjük jellegük szerint, Személy (*Person*) vagy Csoport (*Group*), megadhatjuk kezdési és végdátumukat (*begin date, end date*). Személyeknél ez a születési és halálozási dátum, csoportoknál megalakulási valamint feloszlási dátum. Az előadóknak tetszőleges számú kiadványuk lehet, valamint tetszőleges számú más előadókkal, kiadványokkal és zeneszámokkal és URL-ekkel lehetnek kapcsolatban.

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://musicbrainz.org/ns/mmd-1.0#">
    <artist id="c0b2500e-0cef-4130-869d-732b23ed9df5" type="Person">
        <name>Tori Amos</name>
        <sort-name>Amos, Tori</sort-name>
        <life-span begin="1963-08-22"/>
        <release-list>
            <!-- releases omitted -->
        </release-list>
    </artist>
</metadata>
```

39. példa: Egy előadó leírása

A Release (kiadvány) osztály: Minden kiadványnak van címe (*title*), egy vagy több zeneszáma (*tracks*), típusa (album, feldolgozás (*compiation*), kislemez (*Single*), stb…), státusza (hivatalos, promóció) és nyelvi információ. Tartozhat hozzá kiadványi információ (*release information*). Kapcsolatokkal (*relations*) köthetjük más kiadványokhoz, előadókhoz, zeneszámokhoz, URL-ekhez.

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://musicbrainz.org/ns/mmd-1.0#">
    <release id="02232360-337e-4a3f-ad20-6cdd4c34288c" type="Album
Official">
        <title>Little Earthquakes</title>
        <text-representation language="ENG" script="Latn"/>
        <asin>B000002IT2</asin>
        <artist id="c0b2500e-0cef-4130-869d-732b23ed9df5" type="Person">
            <name>Tori Amos</name>
            <sort-name>Amos, Tori</sort-name>
            <life-span begin="1963-08-22"/>
        </artist>
        <release-event-list>
            <event date="1992-01-13" country="GB"/>
            <event date="1992-01-17" country="DE"/>
            <event date="1992-02-25" country="US"/>
        </release-event-list>
        <disc-list>
            <disc id="ILKp3.bZmvoMO7wSrq1cw7WatfA-"/>
            <disc id="ejdrdtX1ZyvCb0g6vfJejVaLIK8-"/>
            <disc id="Y96eDQZbF4Z26Y5.Sxdbh3wGypo-"/>
        </disc-list>
        <track-list count="12"/>
    </release>
</metadata>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<metadata xmlns="http://musicbrainz.org/ns/mmd-1.0#">
    <track id="d6118046-407d-4e06-a1ba-49c399a4c42f">
        <title>Silent All These Years</title>
        <duration>253466</duration>
        <puid-list>
            <puid id="c2a2cee5-a8ca-4f89-a092-c3e1e65ab7e6"/>
            <puid id="db5b66b3-fa97-4cfa-9296-de7e57ef05f4"/>
            <puid id="fbb3d1b6-f9e7-49ed-834c-03e9ec9726e5"/>
            <puid id="0e2e66e3-13d7-4138-af90-5ec8a4c2db99"/>
            <puid id="4778d58b-50fe-4004-8cd6-462a816114c8"/>
            <puid id="88541d46-9b74-4835-b8e8-d985fc77e02a"/>
            <puid id="42ab76ea-5d42-4259-85d7-e7f2c69e4485"/>
        </puid-list>
    </track>
</metadata>
```

információk megadására. Tartozhat hozzájuk kód (*code*), ország (*country*), megalakulási (founding) és feloszlási (dissolving) idő. A címkék tetszőleges számú címkével, előadóval, kiadvánnyal, URL-el lehetnek kapcsolatban.

Lista elemek (List elements): A metaadat elemek tartalmazhatnak különféle lista elemeket, ilyenek az előadó-lista (*artist-list*), kiadvány-lista (*release-list*), zeneszám-lista (*track-list*), PUID-lista (*puid-list*), kapcsolat-lista (*relation-list*). A fenti, 36. példa tartalmaz egy PUID-listát.

A kapcsolatok leírására tekintsük az alábbi példát, mely egy előadója kapcsolat fennállását szemlélteti egy zeneszám és egy előadó között.

```
...
            <relation type="Performer" target=
               "0fcae62b-906c-4e61-b036-ed9c8fe8dd57">
                <track id="0fcae62b-906c-4e61-b036-ed9c8fe8dd57">
                    <title>Blue Skies (feat. Tori Amos)</title>
                    <duration>306040</duration>
                </track>
            </relation>
...
```

42. példa: Kapcsolat leírása

Az adatbázist böngészhetjük webes felületen keresztül, vagy valamilyen kliensprogramot használva. A metaadatok automatikus feldolgozás céljából HTTP kérésekkel kérdezhetőek le, vagy akár módosíthatóak is egy XML alapú webszolgáltatáson keresztül.

Az URL séma

Minden MusicBrainz objektum (mint pl. előadó, kiadvány, stb.) egy erőforrás. Az erőforrások mind rendelkeznek egy-egy egyedi URL-el, mely egyértelműen azonosítja őket. Minden erőforrás egy kollekció (*collection*) része, mely egy speciális erőforrás, egy bizonyos típus minden objektumát reprezentálja. A webszolgáltatás részére a `http://musicbrainz.org/ws/1/` névtér van lefoglalva.

A névtér a következő szerkezetű:

2. tábla: MusicBrainz URL séma	
http://musicbrainz.org/ws/1/artist/	Előadók kollekciója
http://musicbrainz.org/ws/1/artist/MBID	Egy előadó
http://musicbrainz.org/ws/1/release-group/	Kiadványi csoportok kollekciója
http://musicbrainz.org/ws/1/release-group/MBID	Egy kiadványi csoport
http://musicbrainz.org/ws/1/release/	Kiadványok kollekciója
http://musicbrainz.org/ws/1/release/MBID	Egy kiadvány
http://musicbrainz.org/ws/1/track/	Zeneszámok kollekciója
http://musicbrainz.org/ws/1/track/MBID	Egy zeneszám
http://musicbrainz.org/ws/1/label/	Címkék kollekciója
http://musicbrainz.org/ws/1/label/MBID	Egy címke
http://musicbrainz.org/ws/1/tag/	Tegek
http://musicbrainz.org/ws/1/rating/	Értékelések

(forrás: http://musicbrainz.org/doc/XML_Web_Service)

Kétféleképpen férhetünk hozzá az adatbázisban tárolt metaadatokhoz. Az egyik, ha ismerjük az objektum MBID-jét (egy globális egyedi azonosító, mellyel az adatbázis minden objektuma rendelkezik), közvetlenül lekérdezhetjük az erőforrást. Például a következő URL-t használva, mely Tori Amos-t azonosítja.

```
http://musicbrainz.org/ws/1/artist/c0b2500e-0cef-4130-869d-
732b23ed9df5?type=xml
```

A másik módszer, hogy használjuk valamelyik kollekciót. Mivel ezek a kollekciók rendkívül nagyok, az egész lekérdezése erőforrásigényes és felesleges is lenne, ezért ún. filtereket (szűrőket) is meg kell adnunk, melyek valamely szempont szerint szűrik az adatokat. Például a Tori Amos nevű előadókat a következő filtert használó kéréssel kaphatjuk meg:

```
http://musicbrainz.org/ws/1/artist/?type=xml&name=Tori+Amos
```

A HTTP metódusaival létrehozhatunk (PUT), lekérhetünk (GET), módosíthatunk (POST) és törölhetjük az erőforrásokat (DELETE).

A választ egyszerű XML formában kapjuk meg. A fenti kérdésre a következő választ kapnánk:

```
<?xml version="1.0" encoding="UTF-8"?>
<metadata xmlns="http://musicbrainz.org/ns/mmd-1.0#"
        xmlns:ext="http://musicbrainz.org/ns/ext-1.0#">

<metadata>

   <artist-list offset="0" count="59">

     <artist type="Person" id="c0b2500e-0cef-4130-869d-732b23ed9df5"
            ext:score="100">
           <name>Tori Amos</name>
           <sort-name>Amos, Tori</sort-name>
           <life-span begin="1963-08-22"/>
     </artist>
     <artist type="Person" id="9973d77e-bbcd-4977-86ad-8c300417aa00"
            ext:score="42">
           <name>Tori</name>
           <sort-name>Tori</sort-name>
     </artist>
...
   </artist-list>
</metadata>
```

6 Összefoglalás

A Weben található információk jelenleg is óriási mennyiségűek és hatalmas iramban növekednek. Ezért mindinkább szükség van olyan technológiákra, melyekkel megvalósítható az adatok összekapcsolása, rendszerezése, a hatékonyabb keresés és feldolgozás érdekében. Hogy ezeket a célokat elérjük, a gépeket képessé kell tenni az információk megértésére. Ennek egyik eszköze, hogy az adatokat háttérinformációkkal látjuk el, melyeket metaadatoknak nevezünk. A metaadatok használata univerzális eszköz arra, hogy jelentést társítsunk erőforrásokhoz, és összekapcsoljuk azokat más erőforrásokkal. Másik fontos dolog, mely szükséges az információk helyes feldolgozásához, hogy a gépeket meg kell „tanítanunk" a különböző forrásokból származó részinformációk kombinálására, és hogy ezekből a bonyolultabb összefüggésekből, képesek legyenek következtetéseket levonni, újabb, magasabb szintű információkat előállítani. A szemantikus web pontosan a fenti elképzelést hivatott megvalósítani.

Diplomamunkám célja az volt, hogy bemutassam a „szemantikus web elképzelést", a hozzá kapcsolódó technológiákat, valamint azok felhasználási lehetőségeit. Bemutattam annak réteges architektúráját, részletes leírást adtam a fontosabb technológiákról. Ilyenek:

- Az XML általános célú leíró nyelv, melynek elsődleges célja strukturált szöveg és információ megosztása az Interneten keresztül, számítógép számára is feldolgozható formában. Sok más felhasználási lehetősége mellett, a szemantikus web technológiáinak is alapját képezi.
- Az RDF keretrendszer, mellyel metainformációt társíthatunk webes erőforrásokhoz.
- Az OWL ontológianyelv, amivel formálisan le tudjuk írni egy szakterület alapfogalmainak valamint ezek összefüggéseinek definícióit.

Ezután bemutattam mindezek felhasználási lehetőségét, valamint néhány jelenleg is működő rendszert, mely a szemantikus web technológiáira épül.

Következtetésként elmondható, hogy bár ezeket a technológiákat egyszerűnek tervezték, néhol mégis túl bonyolultak lettek. Jól példázza ezt, hogy a cégek és szervezetek gyakran kidolgozzák saját metaadat formátumukat, mely a céljaiknak jobban megfelel (pl. MusicBrainz MMB), vagy csak egy kisebb részhalmazt használnak (pl. Adobe XMP). Ahhoz, hogy a szemantikus web megvalósuljon, az kellene, hogy minden weboldalhoz le legyen

tárolva minden metainformáció. Ehhez azonban még nincs elég eszköz, mely jól használható, automatizálható lenne. A jelenlegi HTML szabványok még nem is tudnak mit kezdeni a szemantikus tartalommal, használatuk nem teljes mértékben támogatott. A szemantikus keresők szintén korai állapotban vannak. A technológiák fejlődésével és terjedésével egyre közelebb kerülünk a szemantikus web kialakulásához, és ez lehet a Web következő nagy generációja a Web 3.0. Talán a közeljövőben megvalósulhat az „elképzelés”, és könnyebbé válnak mindennapjaink.

Köszönettel tartozom témavezetőmnek, Adamkó Attilának, az iránymutatásért és segítségért.

7 Irodalomjegyzék

Allemang, D. (2008). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL* . Burlington: Morgan Kaufmann.

Az OWL Web Ontológia Nyelv – Alkalmazási esetek és követelmények. (2004). Forrás: http://www.w3c.hu/forditasok/OWL/REC-webont-req-20040210.html

Az OWL Web Ontológia Nyelv – Áttekintés. (2004). Forrás: http://www.w3c.hu/forditasok/OWL/REC-owl-features-20040210.html

Az OWL Web Ontológia Nyelv – Útmutató. (2004). Forrás: http://www.w3c.hu/forditasok/OWL/REC-owl-guide-20040210.html

Az RDF alapfogalmai és absztrakt szintaxisa. (2005). Forrás: http://www.w3c.hu/forditasok/RDF/REC-rdf-concepts-20040210.html

Az RDF bevezető tankönyve. (2005). Forrás: http://www.w3c.hu/forditasok/RDF/REC-rdf-primer-20040210.html

Az RDF Szókészlet Leíró Nyelv 1.0: RDF Séma. (2005). Forrás: http://www.w3c.hu/forditasok/RDF/REC-rdf-schema-20040210.html

Az RDF/XML szintaxis specifikációja. (2005). Forrás: http://www.w3c.hu/forditasok/RDF/REC-rdf-syntax-grammar-20040210.html

Bradley, N. (2005). *The XML-companion (magyar) Az XML-kézikönyv.* Bicske: Szak K.

Daconta, M. C. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management.* Indianapolis: Wiley Publishing, Inc.

Davis Hunter, J. R. (2007). *Beginning XML, 4th Edition (Programmer to Programmer).* Indianapolis: Wiley Publishing, Inc.

Elliotte Rusty Harold. (2004). *XML in a Nutshell, Third Edition.* Sebastopol CA: O'Reilly Media, Inc.

Extensible Markup Language (XML) 1.0 (Fifth Edition). (2008). Forrás: http://www.w3.org/TR/xml/

Extensible Metadata Platform (XMP). (2010). Forrás: http://www.adobe.com/products/xmp/

Extensible Metadata Platform (XMP) Specification: Part 1, Data and Serialization Model. (2008). Forrás: http://www.adobe.com/devnet/xmp/pdfs/XMPSpecificationPart1.pdf

Extensible Metadata Platform (XMP) Specification: Part 2, Standard Schemas. (2008). Forrás: http://www.adobe.com/devnet/xmp/pdfs/XMPSpecificationPart2.pdf

Extensible Metadata Platform (XMP) Specification: Part 3, Storage in Files. (2008). Forrás: http://www.adobe.com/devnet/xmp/pdfs/XMPSpecificationPart3.pdf

Grigoris Antoniou, F. v. (2008). *A Semantic Web Primer, 2nd Edition (Cooperative Information Systems).* Massachusetts: Massachusetts Institute of Technology.

Iván, H. (2003). *Szemantikus Web: egy rövid bevezetés*. Forrás: W3C: http://www.w3.org/2006/Talks/0318-Budapest-IH/

John Hebeler, M. F. (2009). *Semantic web programming.* Indianapolis: Wiley Publishing, Inc.

Lacy, L. W. (2005). *Owl: Representing Information Using the Web Ontology Language.* United Kingdom: Trafford Publishing, Inc.

Last.FM. (2010). Forrás: http://www.last.fm/api/intro

Music Ontology Specification. (2010). Forrás: http://musicontology.com/

Music Ontology Wiki. (2010). Forrás: http://wiki.musicontology.com/index.php/Main_Page

MusicBrainz. (2010). Forrás: http://musicbrainz.org/

Namespaces in XML 1.0 (Third Edition). (2009). Forrás: http://www.w3.org/TR/xml-names/

OWL Web Ontology Languaguage - Use Cases and Requirements. (2004). Forrás: http://www.w3.org/TR/2004/REC-webont-req-20040210/

OWL Web Ontology Languaguage Guide. (2004). Forrás: http://www.w3.org/TR/2004/REC-owl-guide-20040210/

OWL Web Ontology Languaguage Overview. (2004). Forrás: http://www.w3.org/TR/2004/REC-owl-features-20040210/

Péter, J. (2006). *Böngészés a Szemantikus Weben.* Forrás: http://www.inf.unideb.hu/kutatas/gybin/gybin07/Jeszenszky_Peter.pdf

Powers, S. (2003). *Practical RDF.* Sebastopol CA: O'reilly & Associates, Inc.

RDF Primer. (2004). Forrás: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

RDF Vocabulary Description Language 1.0: RDF Schema. (2004). Forrás: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

RDF/XML Syntax Specification (Revised). (2004). Forrás: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

RDFa in XHTML: Syntax and Processing. (2008). Forrás: http://www.w3.org/TR/rdfa-syntax/

RDFa Primer. (2008). Forrás: http://www.w3.org/TR/xhtml-rdfa-primer/

Resource Description Framework (RDF). (2004). Forrás: http://www.w3.org/RDF/

Resource Description Framework (RDF): Concepts and Abstract Syntax. (2004). Forrás: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

Semantic Web. (2010). Forrás: W3C: http://www.w3.org/standards/semanticweb/

Szeredi Péter, L. G. (2005). *A szemantikus világháló elmélete és gyakorlata.* Budapest: Typotex.

The Friend of a Friend (FOAF) project. (2010). Forrás: http://www.foaf-project.org/

Tibor, G. (2005). *Szemantikus web : bevezetés a tudásalapú internet világába.* Budapest: ComputerBooks.

Toby Segaran, C. E. (2009). *Programming the semantic web.* Sebastopol CA: O'Reilly Media, Inc.

URN. (2008). Forrás: http://www.niif.hu/projektek/middleware/urn

Walsh, N. (1998). *A Technical Introduction to XML*. Forrás: http://www.xml.com/pub/a/98/10/guide0.html

XML 10 pontban. (2004). Forrás: http://www.w3c.hu/forditasok/XML_10_pontban.html

XML Base (Second Edition). (2009). Forrás: http://www.w3.org/TR/xmlbase/

XML Information Set (Second Edition). (2004). Forrás: http://www.w3.org/TR/xml-infoset/

XML Information Set Requirements. (1999). Forrás: http://www.w3.org/TR/NOTE-xml-infoset-req

8 Függelék

Program specifikáció

A program célja:

A célom olyan weboldal fejlesztése volt, ahol a zenekedvelők megtalálhatnak minden információt kedvenc zenészeikről, albumaikról vagy zeneszámaikról.

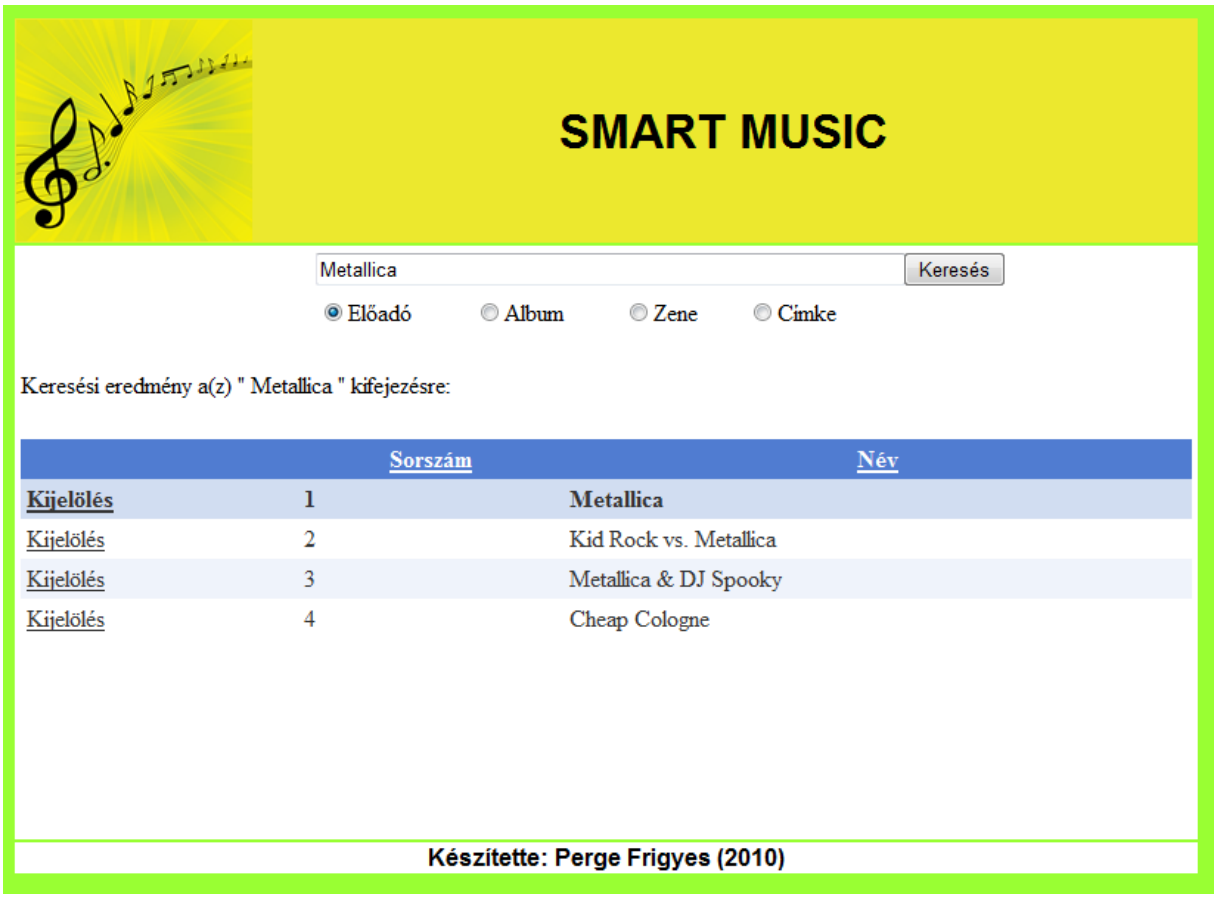
Megvalósítás:

- ASP.NET és C#
- MusicBrainz adatbázisából kérdezem le az adatokat, a MusicBrainz webszolgáltatásán keresztül. (alapvető metaadatokat, pl. album címe, előadója, zeneszámok, stb…)
- Ezeket további információkkal egészítem ki a Last.FM adatbázisából. (pl. borítókép)

Jellemzők és képernyőképek:

A felhasználók tudnak keresni előadó, zeneszám, album, címke alapján.

Keresési eredmények:



Előadóra MusicBrainz-en a következő módon lehet keresni:

```
using MusicBrainz;

string sExpression = Server.HtmlEncode(Request.QueryString["sexp"]);

Query<Artist> artistsRes = Artist.Query(sExpression);
```

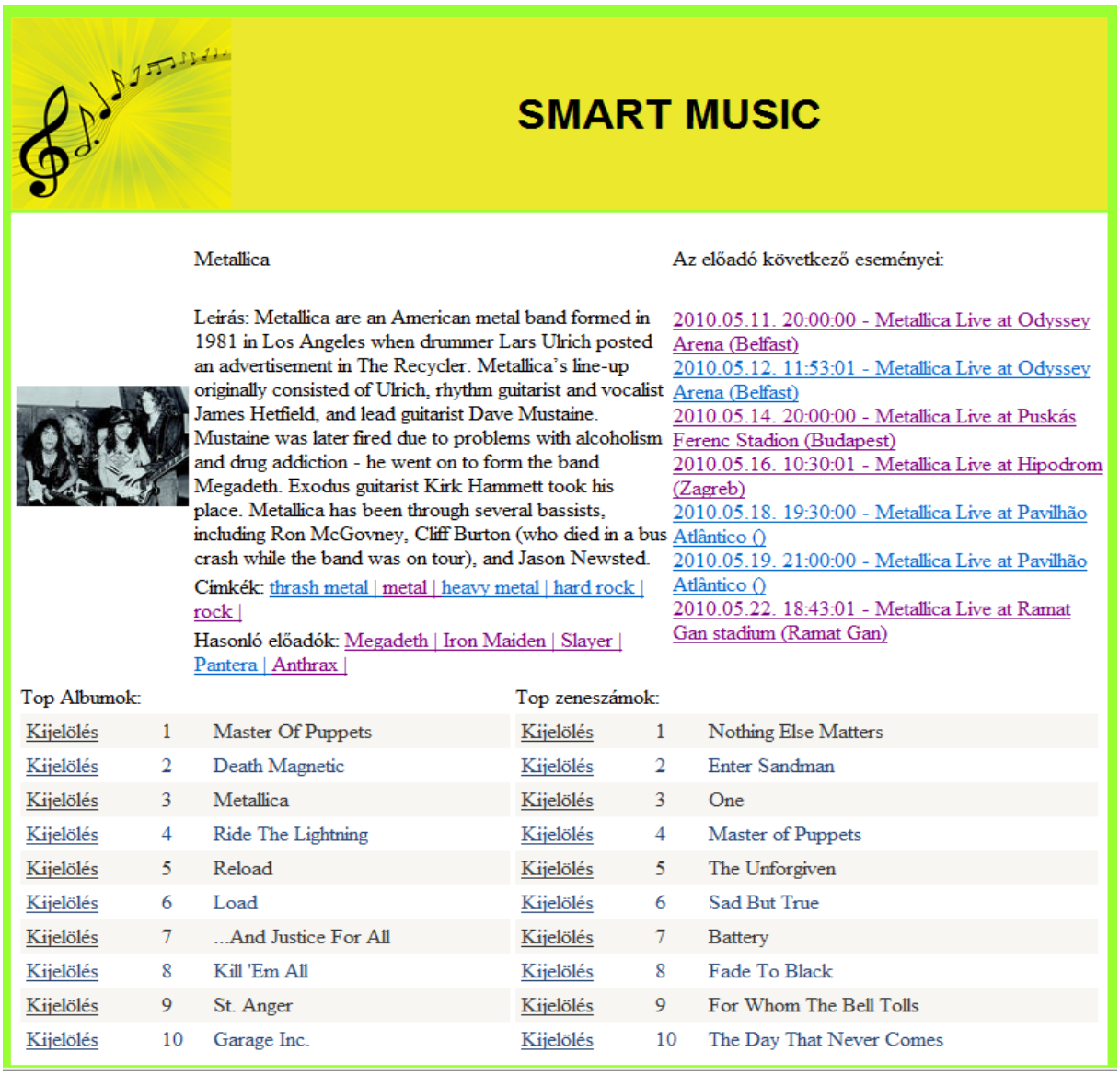
A keresési eredmény megjelenítéséről az alábbi kódrészlet gondoskodik:

```
int index = 0;
foreach (Artist art in artistsRes)
{
    dtResults.Rows.Add(++index, art.GetName());
}
gvResults.DataSource = dtResults;
gvResults.DataBind();
```

Előadó részletei:

Az előadókról megjelenő fontosabb részletek:

- kép, leírás, következő eseményei, a 10 legnépszerűbb albuma, a 10 legnépszerűbb zeneszáma, hasonló előadók, címkék



SMART MUSIC

Metallica

Leírás: Metallica are an American metal band formed in 1981 in Los Angeles when drummer Lars Ulrich posted an advertisement in The Recycler. Metallica's line-up originally consisted of Ulrich, rhythm guitarist and vocalist James Hetfield, and lead guitarist Dave Mustaine. Mustaine was later fired due to problems with alcoholism and drug addiction - he went on to form the band Megadeth. Exodus guitarist Kirk Hammett took his place. Metallica has been through several bassists, including Ron McGovney, Cliff Burton (who died in a bus crash while the band was on tour), and Jason Newsted.

Címkék: thrash metal | metal | heavy metal | hard rock | rock |

Hasonló előadók: Megadeth | Iron Maiden | Slayer | Pantera | Anthrax |

Az előadó következő eseményei:

2010.05.11. 20:00:00 - Metallica Live at Odyssey Arena (Belfast)
2010.05.12. 11:53:01 - Metallica Live at Odyssey Arena (Belfast)
2010.05.14. 20:00:00 - Metallica Live at Puskás Ferenc Stadion (Budapest)
2010.05.16. 10:30:01 - Metallica Live at Hipodrom (Zagreb)
2010.05.18. 19:30:00 - Metallica Live at Pavilhão Atlântico ()
2010.05.19. 21:00:00 - Metallica Live at Pavilhão Atlântico ()
2010.05.22. 18:43:01 - Metallica Live at Ramat Gan stadium (Ramat Gan)

Top Albumok:

Kijelölés	1	Master Of Puppets
Kijelölés	2	Death Magnetic
Kijelölés	3	Metallica
Kijelölés	4	Ride The Lightning
Kijelölés	5	Reload
Kijelölés	6	Load
Kijelölés	7	...And Justice For All
Kijelölés	8	Kill 'Em All
Kijelölés	9	St. Anger
Kijelölés	10	Garage Inc.

Top zeneszámok:

Kijelölés	1	Nothing Else Matters
Kijelölés	2	Enter Sandman
Kijelölés	3	One
Kijelölés	4	Master of Puppets
Kijelölés	5	The Unforgiven
Kijelölés	6	Sad But True
Kijelölés	7	Battery
Kijelölés	8	Fade To Black
Kijelölés	9	For Whom The Bell Tolls
Kijelölés	10	The Day That Never Comes

Például az előadó képét a Last.Fm-ről kérdezem le, majd jelenítem meg az alábbi módon (a Last.Fm kliens inicializálása a Last.Fm webszolgáltatás leírásánál található a 87. oldalon):

```
string mBidArtist = Server.HtmlEncode(Request.QueryString["art"]);
MusicBrainz.Artist artistMB = Artist.Get(mBidArtist);
ArtistInfo a_info = client.Artist.GetInfo(artistMB.Id, "en");
imgArtist.ImageUrl = a_info.ImageLarge.ToString();
```

Album részletei:

Az albumokról megjelenő fontosabb részletek:

- borítókép, leírás, link a megvásárlásához, az albumon található zeneszámok listája, címkék

	Sorszám	Cím	Hossz
Kijelölés	1	Battery	5 : 11
Kijelölés	2	Master of Puppets	8 : 38
Kijelölés	3	The Thing That Should Not Be	6 : 39
Kijelölés	4	Welcome Home (Sanitarium)	6 : 27
Kijelölés	5	Disposable Heroes	8 : 21
Kijelölés	6	Leper Messiah	5 : 42
Kijelölés	7	Orion	8 : 27
Kijelölés	8	Damage, Inc.	5 : 31

Ha ismerjük valaminek, a MusicBrainz azonosítóját direkt módon is lekérdezhetjük:

```
string mBidAlbum = Server.HtmlEncode(Request.QueryString["alb"]);
Release albumMB = Release.Get(mBidAlbum);
```

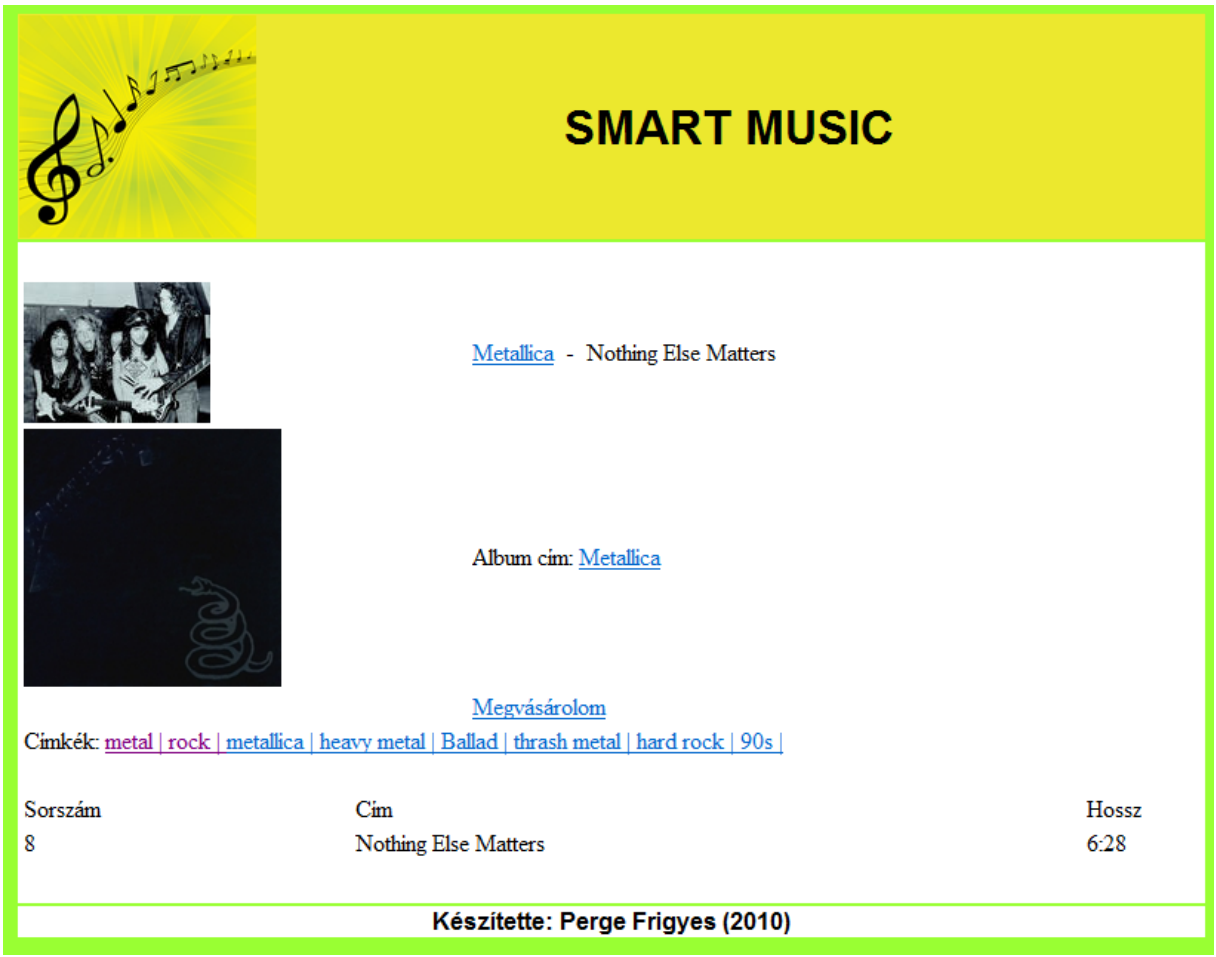
Az album részleteit a következő kódrészlet jeleníti meg.

```
try { lblAlbumTitle.Text = albumMB.GetTitle(); }
catch (Exception ex) { }
try { hlbArtist.Text = albumMB.GetArtist().GetName(); }
catch (Exception ex) { }
try { lblReleaseDate.Text += albumMB.GetEvents()[0].Date; }
catch (Exception ex) { }
```

Zeneszám részletei:

Az zeneszámokról megjelenő fontosabb részletek:

- leírás, link a megvásárlásához, címkék, előadóról kép, albumborító kép



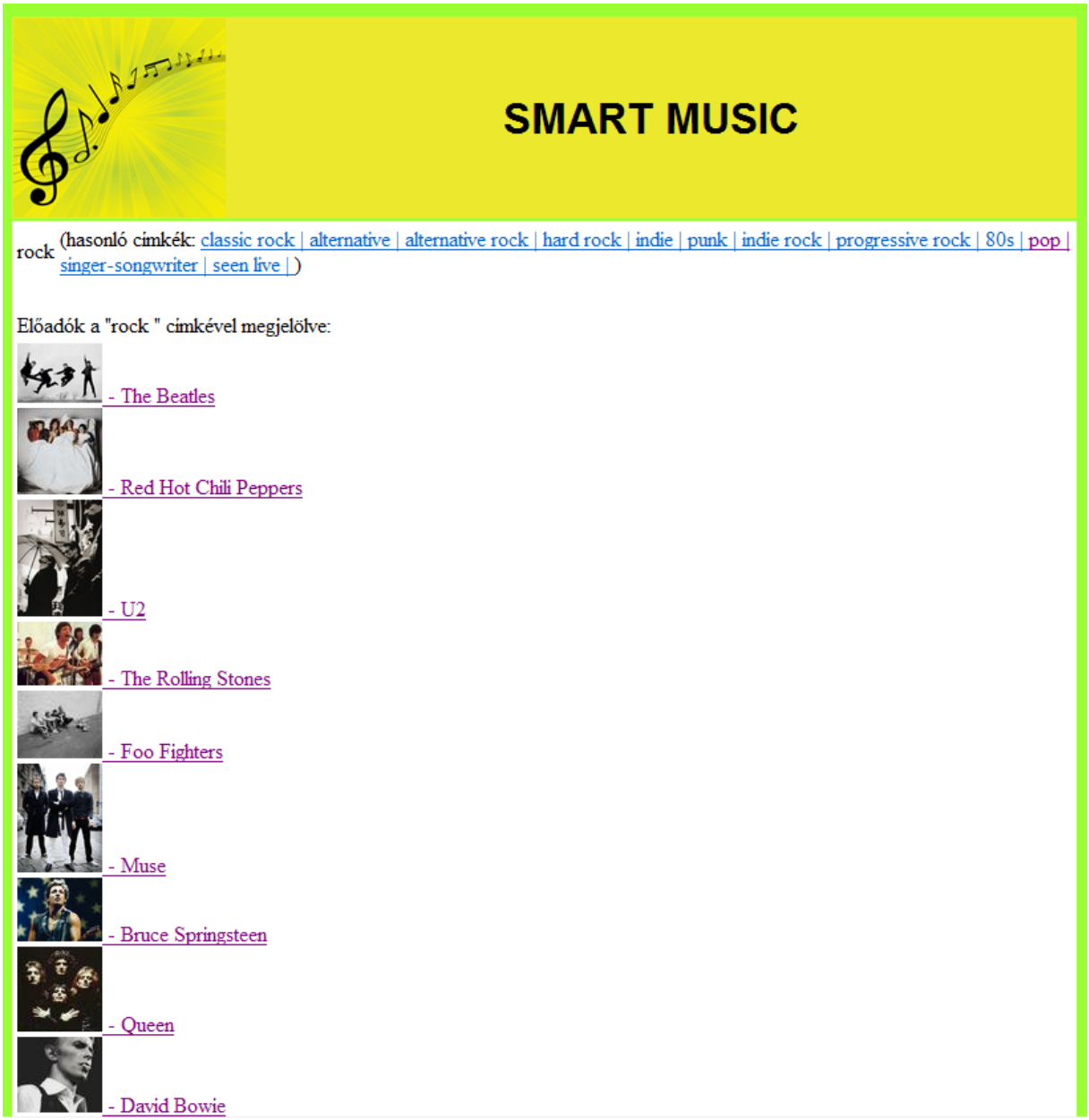
A címkéket a Last.FM adatbázisából kérdezem le, majd adom hozzá az oldalhoz (a Last.Fm kliens inicializálása a Last.Fm webszolgáltatás leírásánál található a 87. oldalon):

```
private void showTags(string artistName, string title)
{
    RankingList<TagInfo> tags = client.Track.GetTopTags(artistName,
        title);
    int i = 0;
    foreach(TagInfo t in tags)
    {
        HyperLink hl = new HyperLink();
        hl.Text = t.Name + " | ";
        hl.NavigateUrl = ("~/TagDetails.aspx"
            + "?" + "tag=" + Server.UrlEncode(t.Name));
        phTags.Controls.Add(hl);
        if (++i > 7) break;
    }}
```

Címke részletei:

A címkékről megjelenő fontosabb részletek:

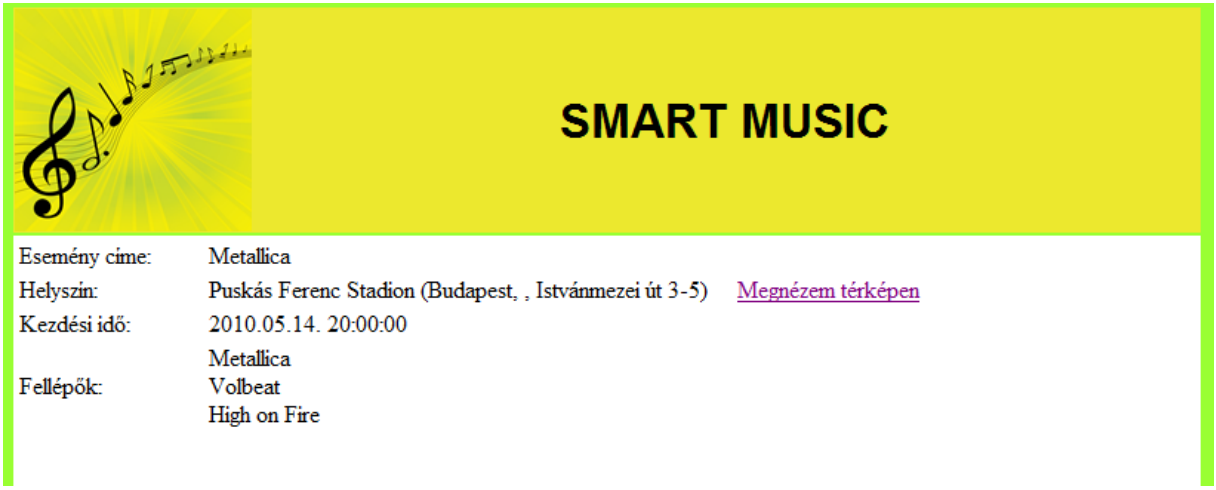
- 10 hasonló címke
- a 10 legnépszerűbb előadó ezzel a címkével
- a 10 legnépszerűbb album ezzel a címkével
- a 10 legnépszerűbb zeneszám ezzel a címkével



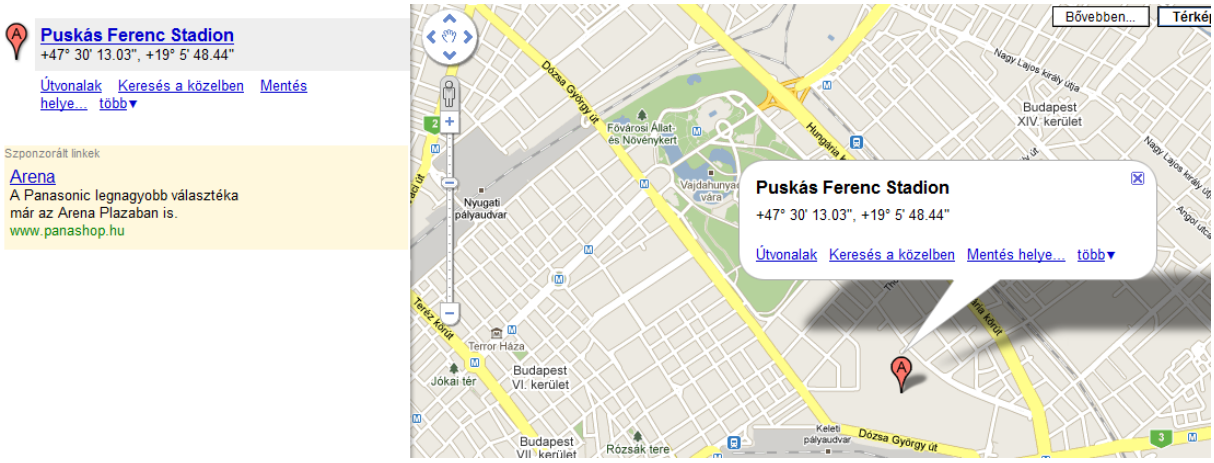
Esemény részletei:

Az eseményekről megjelenő részletek:

- leírás
- a helyszín megnézhető térképen



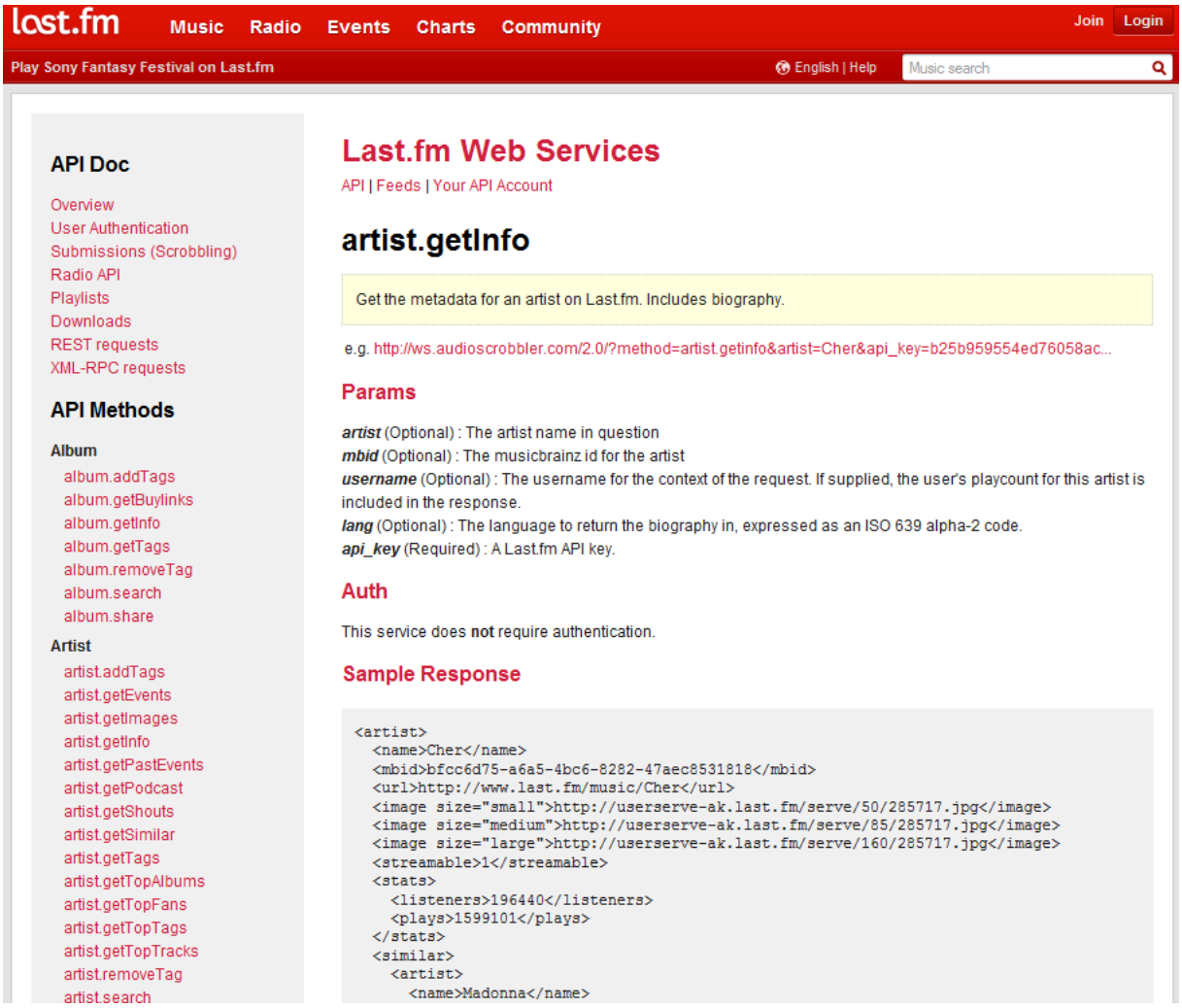
Térképen megjelenítve:



A lekérdezést és megjelenítést végző kódrészlet:

```
int eveId = Int32.Parse(Request.QueryString["eve"]);
MusicEvent eve = client.Event.GetInfo(eveId);
showContent(eve);

private void showContent(MusicEvent eve)
{
    lblName.Text = eve.Title;
    lblVenue.Text = eve.Venue.Name
        + " (" + eve.Venue.City
        + ", " + eve.Venue.Country
        + ", " + eve.Venue.Street + ")"; ...
```

Részlet a Last.Fm webszolgáltatás leírásából:



(forrás: http://www.last.fm/api/show?service=267)

A Last.Fm klienst a következő kódrészlet inicializálja:

```
using LastFmLib;
using LastFmLib.General;
using LastFmLib.API20.Types;
using LastFmLib.API20;

private LastFmClient initLastFmClient()
{
    MD5Hash key = new MD5Hash("73330...1af6", true,
        System.Text.Encoding.UTF8);
    MD5Hash secret = new MD5Hash("3f712...a176", true,
        System.Text.Encoding.UTF8);
    AuthData myauth = new AuthData(key, secret);
    LastFmLib.API20.Settings20.AuthData = myauth;
    LastFmClient client = LastFmClient.Create(myauth);
    return client;}
```